Efficient Simulation of Rare Events in one-dimensional systems using a parallelised cloning algorithm

Brewer, Tobias

Award date:
2019

Awarding institution:
University of Bath

Efficient Simulation of Rare Events in one-dimensional systems using a parallelised cloning algorithm

Tobias Brewer

A thesis submitted for the degree of
Doctor of Philosophy

University of Bath
Department of Physics
October 2018

Access to this thesis/portfolio in print or electronically is restricted until ..... *(date)*.
Signed on behalf of the Doctoral College.....(print name).....

Attribution 4.0 International (CC BY 4.0)

Declaration of any previous submission of the work

The material presented here for examination for the award of a higher degree by research ~~has /~~ has not been incorporated into a submission for another degree.

(If applicable, provide the relevant details i.e. those parts of the work which have previously been submitted for a degree, the University to which they were submitted and the degree, if any, awarded).

Candidate's signature ~~.....~~

Declaration of authorship

I am the author of this thesis, and the work described therein was carried out by myself personally, ~~with the exception of ..... article/chapter where~~ .....0%..... *(detail the amount in percentage terms)* of the work was carried out by other researchers *(e.g. detail any collaborative works included in the thesis in terms of formulation of ideas, design of methodology, experimental work, and presentation of data in journal format).*

Candidate's signature ~~.....~~

Abstract

We consider the population dynamics that are implemented by the cloning algorithm for analysis of large deviations of time-averaged quantities. We consider exclusion processes acting on particles on one-dimensional lattices such as the simple symmetric exclusion process and the Fredkin Process. We use large deviation theory to quantify the probabilities of rare events. To achieve this we adapt a numerical algorithm which employs a combination of biased cloning and simulation of modified dynamics. We establish its accuracy within particular regimes, determine which configurations are likely to produce rare events and quantify the convergence of the algorithm with respect to algorithmic parameters. We investigate the efficiency and speed-up obtained when using different parallelisation techniques to implement the algorithm which involves complex communication patterns between systems.

Acknowledgments

I would like to thank Dr. Robert L. Jack, Dr Stephen R. Clark and Dr Russell Bradford for their support throughout my PhD and for their contributions to the research paper [19] that we wrote together. This paper included results collected by me as well as contributions to writing the text from all four of us. I would also like to thank them for the feedback that they provided throughout writing this thesis which includes text written and results collected by me. Thankyou also to Dr Juan P. Garrahan at the University of Nottingham for the ideas he suggested for our research on Fredkin Chains. Thankyou also to my friends and family for their support. This research made use of the Balena High Performance Computing (HPC) Service at the University of Bath. Thankyou also to ClusterVision for providing and funding my studentship and for their help and assistance throughout my PhD.

Contents

Page

1 Introduction **7**

1.1 Statistical Mechanics, Equilibrium and Non-Equilibrium 9

1.2 Examples of Phase Transitions 11

1.3 New Contributions 13

2 Large Deviation Theory and the Cloning Algorithm **16**

2.1 Large Deviation Theory for Time Averaged-Quantities 17

2.2 Modified Dynamics 19

2.3 Cloning Algorithm 21

2.4 Computational Procedures 23

2.5 Estimating Averages with Respect to $\tilde{P}$ 25

3 Dynamical Phase Transition in the SSEP **27**

3.1 Model and Choice of Dynamical Observable 28

3.2 Theoretical Analysis of Dynamical Phase Transition 30

3.3 Properties of Dynamical Phase Transition 31

3.4 Scaling at the Dynamical Phase Transition 36

3.5 SSEP Summary 37

4 Algorithm Performance in the SSEP **39**

4.1 Clone Selection Methods 40

4.2 Effect of Cloning Method and Choice of Δt 41

4.3 Convergence 43

4.3.1 Time t_{obs} Convergence 43

4.3.2 Population Size n_c Convergence 49

4.4 Performance Summary 53

5 Large Deviations in Fredkin Processes **54**

5.1 Fredkin Process and Relevant Observables 54

5.2 Large Deviations in Activity (Hops) . . . 58
5.2.1 Peak in Susceptibility . . . 61
5.3 Large Deviations in Area Beneath the Dyck Path . . . 64
5.4 Algorithm Performance . . . 65
5.4.1 Measuring Activity (Hops) . . . 66
5.4.2 Measuring Area Beneath the Dyck Path . . . 73
5.5 Fredkin Summary . . . 77

6 Computational Approaches for Measuring and Improving Speed and Efficiency 79
6.1 Serial Code and Definitions . . . 80
6.2 OpenMP Code . . . 81
6.3 MPI Code . . . 82
6.3.1 MPI *Pack* Functions . . . 83
6.4 Reduced Communications . . . 84
6.4.1 Communication Patterns . . . 85
6.5 Non-Blocking Implementations . . . 86
6.5.1 Test Codes . . . 87
6.5.2 Packing Multiple Systems . . . 92
6.6 Hybrid Implementations . . . 93
6.6.1 Test Code . . . 95
6.7 Weak Scaling . . . 96
6.8 Derived Data Types . . . 97
6.9 Compilers and Optimisers . . . 98
6.10 Computation Summary . . . 99

7 Conclusions and Future Work 100
7.1 Conclusions . . . 100
7.2 Future Work . . . 108

A Relating $\psi(s)$ to $\pi(a)$ Using the Observable Value at time t 110

B Calculation of Hop Success Rates 113

C Satisfying the Master Equation 116

D Path Measures 118

E Derived Data Type for a One-Dimensional Lattice 121

List of Tables

Page

4.1 Relative Error in Activity when Varying the Observation Time t_{obs} in the SSEP 45
4.2 Normalisation Coefficients $c_{\mathcal{O}}(t')$ of Several Observables in SSEP . . . 48
4.3 Relative Error in Activity when Varying the Number of Systems n_{c} in the SSEP 50

5.1 Relative Error in Activity when Varying the Observation t_{obs} in Fredkin Processes when Biasing Activity 69
5.2 Normalisation Coefficients $c_{\mathcal{O}}(t')$ of Various Observables in Fredkin Processes when Biasing Activity 71
5.3 Relative Error in Activity when Varying the Number of Systems n_{c} in Fredkin Processes when Biasing Activity 72
5.4 Relative Error in the Area Beneath the Dyck Path when Varying the Observation Time t_{obs} in Fredkin Processes when Biasing the Area Beneath the Dyck Path 75
5.5 Relative Error in the Area Beneath the Dyck Path when Varying the Number of Systems n_{c} in Fredkin Processes when Biasing the Area Beneath the Dyck Path 77

6.1 Set of Parameters for a Test Case at which we Obtain Results. 80
6.2 Run Times Obtained by MPI implementations with Different Amounts of Packing 84
6.3 Run Times Obtained by MPI implementations with and without Reduced Communications. 85
6.4 Run Time of Test Code A 88
6.5 Run Times of the Test Code which Copies One Component of Length 100 when the Request Array of the WaitAny Function is Split. 91

List of Figures

Page

3.1 Example of SSEP on a One-Dimensional Lattice 30
3.2 Sample Trajectories of SSEP 32
3.3 The Mean Square Value of the First Fourier Component of Density . 33
3.4 Results Obtained from our Implementation of the Algorithm for the SSEP 34

4.1 Number Line Used in Clone Selection Methods 40
4.2 Dependence of the Algorithm's Results on the Size of the Cloning Interval and the Clone Selection Mechanism 42
4.3 Time Convergence with Respect to t_{obs} 44
4.4 The p_{ave} Measure of Variables Around the Final Transient Regime . . 46
4.5 Autocorrelation Function of Variables with Respect to the Final Time t_{obs} 48
4.6 Clone Convergence with Respect to n_{c} 50
4.7 p_{ave} and p_{end} Distributions of Activity Per Cloning Interval K^{β} 51

5.1 An Example of a Dyck Path and its Corresponding Dyck Word . . . 56
5.2 Illustration of a Fredkin Process on a One-Dimensional Lattice 56
5.3 Large Deviation Function and Directly Measured Activity when Biasing Activity in the Fredkin Process 59
5.4 Large Deviation Function and Directly Measured Area from Algorithm and Theory when Biasing Activity in the Fredkin Process . . . 60
5.5 Sample Trajectories of Fredkin Processes when Biasing Activity . . . 61
5.6 Density Profile of Fredkin Processes when Biasing Activity 62
5.7 Directly Measured Area Beneath the Dyck Path in Fredkin Processes when Biasing the Activity 63
5.8 Directly Measured Centre of Mass in Fredkin Processes when Biasing the Activity 63

5.9 Large Deviations, Activity and Susceptibilty of Fredkin Processes when Biasing Activity. 64
5.10 Large Deviation Function and Directly Measured Area Beneath the Dyck Path when Biasing the Area Beneath the Dyck Path 65
5.11 Sample Trajectories of Fredkin Processes when Biasing the Area Beneath the Dyck Path 66
5.12 Density Profile of Fredkin Processes when Biasing the Area Beneath the Dyck Path 67
5.13 Centre of Mass of Fredkin Processes when Biasing the Area Beneath the Dyck Path 68
5.14 Time Convergence of the Algorithm when Biasing Activity in Fredkin Processes 68
5.15 Clone Convergence of the Algorithm when Biasing Activity in Fredkin Processes 69
5.16 Time Convergence of the Algorithm when Biasing Activity in Fredkin Processes Near the Maximum Susceptibility 70
5.17 The p_{ave} Measure of Variables Around the Final Transient Regime in Fredkin Processes when Biasing the Activity. 71
5.18 Autocorrelation Function of Variables with Respect to the Final Time t_{obs} in Fredkin Processes when Biasing the Activity. 72
5.19 Clone Convergence of the Algorithm when Biasing Activity in Fredkin Processes Near the Maximum Susceptibility 73
5.20 p_{ave} and p_{end} Distributions of Activity Per Cloning Interval K^{β} in Fredkin Processes 74
5.21 Time Convergence of the Algorithm when Biasing the Area Beneath the Dyck Path in Fredkin Processes 75
5.22 Clone Convergence of the Algorithm when Biasing the Area Beneath the Dyck Path in Fredkin Processes. 76

6.1 Illustration of the Serial Implementation of the Algorithm 80
6.2 Illustration of the OpenMP Implementation of the Algorithm 82
6.3 Illustration of the MPI implementation of the algorithm 83
6.4 An Example of Reduced MPI Communication 85
6.5 MPI Communications Pattern Under Simple Communications and Reduced Communications 86
6.6 Fixed Communications Pattern for Test Codes. 89
6.7 Run Time of Test Code B. 90
6.8 Run Time of Test Code which Copies One Component of Length 100. 91

6.9 Run Time of the Test Code which Uses Multiple Request Arrays . . . 92
6.10 Run Times of the Code with System-Packing with Blocking and with Non-Blocking Communications. 93
6.11 Illustration of the Hybrid Implementation of the Algorithm 94
6.12 Run Times of the Hybrid Code. 95
6.13 Run Time of the Test Code where the Length of the Component Being Copied is Varied. 97
6.14 Weak Scaling of the Run Time of the Code 98

Chapter 1

Introduction

Rare events are important across scientific fields from geology [52] to climate dynamics [40]. In particular, there has been recent interest in rare events where a *time-averaged* observable quantity has a non-typical value [35, 61, 110]. If the system is ergodic and the average is taken over a long period then these rare events can be described by *large deviation theory*. Studies of such rare events have proven useful in glassy materials [69, 48], protein-folding [120, 121, 91] and integrable systems [112].

Large deviation theory was first developed in the field of insurance mathematics by Harald Cramér [32] and Filip Lundberg [31]. Cramér and Lundberg were both actuaries who used large deviation theory to describe the probability of an insurer going into insolvency, this was known as ruin theory. Large deviation theory allows us to approximate the asymptotic behaviour of probability distributions. It has a range of applications [117, 59] such as in finance and insurance [100] and allows us to approximate the probabilities of rare events.

The study of large deviations in simple systems of interacting diffusing particles has led to the exploration of dynamical phase transitions [12] and attempts to describe their properties [13, 85, 14]. Phase transitions are interesting because they correspond to a step change in the value of an observable. To fully understand the physics of a system it is important to be able to understand the properties of this shift and why it occurs. The phase transitions that we consider are between a hyperuniform state [115] and a less easily characterized ordered state [85]. The numerical methods used for evaluating large deviations often exhibit large systematic errors close to these phase transitions [94]. This makes phase transitions a useful place at which to test and improve these methods.

The nature of rare events makes their direct measurement difficult, which has motivated a number of computational methods for studying them [18, 74, 75].

Grassberger introduced a numerical procedure [51] for sampling configurations from a given distribution which simulates a large number of systems and clones/deletes systems which have a particularly large or small weight associated with them. This approach has been developed by Giardinà, Peliti and Kurchan [50] and then by Lecomte and Tailleur [86] leading to a *cloning* algorithm. This method shares some features with other rare-event sampling methods such as forward-flux sampling [1] and weighted-ensemble dynamics [66, 122]. Other methods – particularly transition path sampling [16] – can also be applied to similar problems. Transition path sampling allows the computational study of rare events without prior knowledge of reaction coordinates or transition rates [16]. It has proved particularly efficient in sampling reactive trajectories [76]. Path sampling can be substantially more effective than simply the simulation of dynamics and is particularly well suited to situations when there is a large separation between the timescale of the transition event itself and the preceding period of time [28].

The cloning algorithm has been applied to a range of systems [112, 101, 67, 96]. Examples of systems in which we may simulate the dynamics to investigate rare events are molecular dynamics [29, 23], supercooled liquids [77] and rogue waves [98] [49]. The method gives powerful results but requires the simulation of a large number of instances of the systems and only produces accurate results in the limit of the number of systems tending to infinity. Nemoto, Hidalgo and Lecomte [95, 64] and Ray, Chan and Limmer [103] analyse the systematic and random errors that occur when the algorithm is executed with a finite number of systems. The scaling of these errors has been determined by an analytical description and verified using a numerical approach which measures the speed at which an estimator converges towards its true value [95, 64]. To improve the efficiency of this convergence some versions of the algorithm use controlling forces or simple guiding models [96, 103, 94].

In some previous analyses of the convergence of the algorithm it was often assumed that the clone population is larger than the total number of states visited by the model [95, 64]. This is not generally the case in typical applications so we analyse here the case where the number of clones is much smaller than the total number of states that the model allows the system to exist in. The algorithm requires a large time and only produces accurate results in the limit of the observation time tending to infinity. As the observation time increases the systematic error decreases. In *Part II* of their *Finite Scalings* paper [64], Hidalgo, Nemoto and Lecomte investigated the scaling of these errors.

1.1 Statistical Mechanics, Equilibrium and Non-Equilibrium

Many of the ideas that the cloning algorithm is based on are related to ideas in statistical mechanics. Statistical mechanics is based on the idea that a system can exist in a discrete set of microstates which define the values of all possible microscopic variables. It is used to study physical properties of macroscopic systems with a large number of degrees of freedom. In the context of statistical mechanics, internal parameters are those which depend on the positions and momenta of particles entering the system examples of which are internal energy, temperature and pressure. External parameters, in this context, are those only determined by the co-ordinates of external bodies interacting with the system such as the electric and magnetic field strengths and the volume.

The internal parameters of a system can be described as intensive or extensive. Extensive parameters are some sometimes called additive parameters and include quantities like the energy and the entropy. Extensive parameters are proportional to the amount of substance in the system. Intensive parameters are quantities which are not dependent on the amount of substance in the system such as the temperature and pressure. A macroscopic state of the system is described by these parameters and is not related to microscopic parameters.

One of the important concepts in statistical mechanics is the distinction between equilibrium and non-equilibrium. An equilibrium system is one in which conserved quantities do not change even in a very long waiting time. In equilibrium, the system will be in a state that is independent of its past history. Nonequilibrium systems are any systems that cannot be described as being in equilibrium including those in a transient time before reaching equilibrium. Non-equilibrium systems are not well understood in comparison to our understanding of equilibrium systems.

If a system is not in the equilibrium state, the first postulate of thermodynamics states that over some period of time if it is isolated (or finite) it tends towards its equilibrium state. It also says that, macroscopically, this state is characterized completely by the internal energy, the volume and the amounts of the chemical components. This postulate also states that an isolated system only has one intrinsic state, its equilibrium state. Once a system is in its equilibrium state it can only ever leave it spontaneously by an external force being exerted on it.

Phase transitions are transformations of thermodynamic systems from one phase to another where their properties change, sometimes dramatically. In particular environments, systems that are quantified by external parameters such as

temperature, pressure and magnetic field exhibit singularities in their associated thermodynamic variables. Classic examples are the boiling of liquid or a ferromagnet losing its magnetization on reaching its Curie temperature [82]. Phase transitions have been also been observed in various other contexts such as in percolation [65], cosmology [88] and protein folding [119].

These transitions can be described as being first-order phase transitions or second-order phase transitions. The order referred to is the order of the lowest derivative of the free energy for which a step is observed at the phase transition. The phase transition occurs at a critical value and the free energy is a continuous function at the phase transition and is a smooth function in the case of a second-order phase transition. For first-order phase transitions, the derivative of the free energy exhibits a step at the critical value which is also referred to as a transition point. The second derivative therefore displays a singularity. The quantity in which this divergence occurs is referred to as the susceptibility.

In the case of a second-order phase transition, the first derivative does not exhibit a discontinuity at the transition point and is an example of a continuous transition. The first derivative is similar to the function itself in first-order transitions in that it exhibits a kink at the critical value. The second derivative in second-order transitions is therefore where we observe a discontinuity which is represented by a step change in its value at the transition point.

Systems that are out of equilibrium have significant differences to their equilibrium equivalents. In out of equilibrium systems unusual phase transitions can take place [115]. One approach to understand the steady state of non-equilibrium systems is macroscopic fluctuation theory. Macroscopic fluctuation theory is based on the probability of fluctuations in thermodynamic variables and currents. It provides a macroscopic treatment of stationary nonequilibrium states for driven diffusive systems [10].

Here we are particularly interested in dynamical phase transitions observed in systems that exhibit dynamical regimes. Phase transitions are interesting because a very small change in the value of a parameter can lead to a very large change in behaviour. This makes them important for understanding natural processes. In dynamical phase transitions, a quantity plays an analogous role to the free energy in terms of a step occurring in one of its derivatives. The dynamical phase transition terminology began in the 1980's, in a time when the thermodynamic formalism of Ruelle and others was exploited to characterize the dynamical regimes exhibited in simple iterated maps [6, 85].

Models in which dynamical phase transitions have been observed include the

weakly asymmetric exclusion process [12], kinetically constrained models [15] and finite size studies [14]. The cloning algorithm has been used to study dynamical phase transitions in the contact process [86] and in the one-dimensional Fredrickson-Andersen model [96]. Much progress has been made in the theoretical understanding of phase transitions and critical phenomena by the parallel application of many approaches to simple lattice models and their continuum analogs [89]. These approaches include exact solutions, mean-field theories, computer simulations, series expansions and renormalization group methods [89].

1.2 Examples of Phase Transitions

Here we use the cloning algorithm to analyse the simple symmetric exclusion process (SSEP), a simple model of particles hopping on a one-dimensional lattice. The SSEP is used widely for modelling dynamic systems in the fields of physics, chemistry [45] and biology [44] and in theories of stochastic processes [114]. The rare events (large deviations) in this model have been analysed theoretically in [3, 85]. We also consider the Fredkin Process [105, 93] a similar model with additional rules governing transition rates and allowed transitions within the process. This allows direct comparison between the results obtained from the SSEP and the results obtained from the Fredkin Process.

The models that we analyse, the SSEP and the Fredkin Process are examples of lattice models. Lattice models have been important in equilibrium statistical mechanics for understanding phase transitions and critical phenomena. One virtue of lattice models is that they allow the isolation of specific features of the system and for them to be connected with macroscopic properties. Lattice models of nonequilibrium processes that have been studied include driven lattice gases, contact processes [86], traffic models [102] and surface catalytic reactions [89].

The SSEP model is very simple but exhibits a large range of rare-event phenomena, including the assembly of many particles into a large macroscopic cluster. Additionally, the model can exhibit hyperuniform states [115] in which density fluctuations are strongly suppressed. These regimes are separated by a dynamical phase transition [85] at which numerical calculations are difficult and which is hence a useful place to test the algorithm.

The phase transition is known to exist for large systems [85] and the value of its transient point is known. The large deviations of the SSEP are known [85] in the limit of infinitely large systems but for finite systems are only known up to the phase transition [3]. This makes the region above the phase transition an interesting place

to test the algorithm for systems of finite size. Also, the behaviour of the system around the phase transition is not understood, nor the scaling of the position or height of the maximum susceptibility with system size.

The Fredkin Process is interesting because it is similar to the SSEP but acts under additional restrictions. These additional restrictions include occupancy rules which produce long range correlations which we do not observe in the SSEP. The Fredkin Process therefore provides an opportunity to understand how these additional restrictions and long range correlations affect the large deviations of the system and its properties at large deviations. As when we investigate the SSEP, we measure large deviations of the particle activity. As the Fredkin Process model is still a simple process it is straightforward to use it to measure the large deviations of other quantities. It is then of interest to observe whether the large deviations of these other quantities correspond to the same rare events.

In the Fredkin Process, we are again interested in the position and height of the maximum susceptibility, how these values scale with system size and how they are affected by additional restrictions. The restrictions that the Fredkin Process acts under are expected to affect the large deviations of the process and hence the position of the maximum susceptibility. There are also results that have been obtained on the size of the spectral gap of the Fredkin Process [93]. This is of interest because it is related to timescales within the process which can be compared to timescales within the SSEP.

Our study here has four purposes. First, to analyse the finite-size scaling of a dynamical phase transition in the SSEP, associated with a particular class of rare events. Second to understand the large deviations of a Fredkin Process. Third, we analyse the performance of the cloning algorithm, particularly its convergence as a function of population size and observation time. The algorithm is often used to analyse rare events in processes and systems which take a long time to simulate. Researchers look for a quick implementation that uses only a small amount of computational resources. Therefore the fourth purpose of this study is to analyse the performance of the method in terms of speed-up and efficiency when it is parallelised on high-performance computers, using OpenMP and MPI for (CPU-based) computation.

The following chapters begin with chapter 2 which provides background on large deviation theory and details of the cloning algorithm. In chapter 3 we describe the SSEP and its dynamical phase transition and present some numerical results that allow finite size effects to be characterised. We then explain in greater detail in chapter 4 how the algorithm is applied to the SSEP and how the method is optimised

to reduce errors. We also analyse the underlying causes of these errors. In chapter 5 we describe the Fredkin Process and obtain numerical results that demonstrate how properties of the system are related to its rare events. We also discuss the number of systems and units of time that are required when we run the algorithm to keep the errors small and what the causes of these errors are. Chapter 6 discusses the computational implementation and the parallelisation techniques that we have investigated. In chapter 7 we summarise our conclusions and speculate on possible future work that would build on the results that we have obtained.

1.3 New Contributions

We make several new contributions in this thesis. Firstly, we obtain values of the large deviation function and its cumulants for activity in the SSEP around the phase transition in systems of finite size. Previously data around this phase transition has only been obtained for the infinite system size limit and below the phase transition for systems of finite size from theoretical values. Our contribution is therefore to generate values of the large deviation function and its cumulants above the phase transition. These results have only previously been presented in our recent paper [19].

As well as obtaining these mathematical quantities around the phase transition, we also attain and present the properties of the system around the phase transition, specifically those properties relating to the clustering of particles. This includes trajectories depicting the evolution over time of the system on which the SSEP occurs. It also includes the first Fourier component of density and how it varies across the phase transition. These results have also only previously been displayed in our paper [19].

We have also made some new contributions concerning the performance of the algorithm. The dependency of the accuracy of the algorithm on the size of the cloning interval is not fully understood. We compare two statistical methods for deciding which systems are cloned. In each case we measure how the statistical errors associated with our algorithmic results vary with the size of the cloning interval and hence for each size of cloning interval determine which cloning method produces the smallest statistical errors. We also determine how the systematic errors of the algorithmic results change with the size of the cloning interval.

The scaling of errors with respect to the observation time of the algorithm and the number of systems that are simulated is known. Here we verify that these scalings hold for the system sizes that we consider for the SSEP. These scalings

provide an estimate of the true values of the large deviations of activity that would be obtained if the algorithm were run for an infinite observation time and with an infinite number of systems. We use this estimate to determine values for the number of units of time and the number of systems that are needed for the algorithmic values to converge to within a 2% criterion of the estimate of the true value.

We have also measured the evolution over time and autocorrelations of a few observables relating to the SSEP. The timescales associated with these observables are informative for understanding the values of the observable time that are required for the algorithm to converge. We have also obtained distributions of two measures of observables that we are interested in. The overlap of these distributions is useful for understanding how many systems are required for the algorithm to converge.

Several new contributions come from our chapter on the Fredkin Process. Large deviations in the Fredkin Process are not something that have previously been considered in great detail and hence the results that we obtain are additions to the field of large deviations. These results include values of the large deviations in activity in the Fredkin Process, specifically the large deviation function and its cumulants the activity and the susceptibility.

These results allow a direct comparison between the SSEP and the Fredkin Process and for the effect of the restrictions on the SSEP that are enforced in the Fredkin Process to be measured. These comparisons include effects on the mathematical quantities associated with the processes such as the large deviation function, the activity and the susceptibility. Furthermore they include the effects of the restrictions on the trajectories of the systems. We also consider how many units of time and how many systems are required for the algorithm to converge when simulating the Fredkin Process and compare them to the corresponding required values for the SSEP.

The configurations generated by our simulations of the Fredkin Process can be described by diagrams called Dyck Paths. Each Dyck Path has an area associated with it. In addition to generating results for large deviations in activity in the Fredkin Process, we also produce results for large deviations in the area beneath the Dyck Path. This allows us to make a comparison between the rare events that we observe when we bias the activity in the Fredkin Process with when we bias the area beneath the Dyck Path. It also allows us to make a comparison between the observable values that we measure in each case.

We also state the optimal high performance computing (HPC) techniques to use for the algorithm code to run as quickly and efficiently as possible. There has not previously been literature that has stated the relative performance of different

computational implementations in executing the specific cloning algorithm that we are studying. The HPC techniques for which we obtain results are parallelisation using MPI and OpenMP which we directly compare with a serial code and parallelisation using an MPI-OpenMP hybrid code. Our results for parallelising the code using MPI include results for blocking MPI implementations and non-blocking MPI implementations.

Chapter 2

Large Deviation Theory and the Cloning Algorithm

In this chapter we describe large deviation theory [116, 117, 92, 41] and how a cloning algorithm can be used on a large population of systems to focus on the rare events of interest. We discuss relevant theoretical and computational methods from the literature and adapt them to our requirements. The content in this chapter and chapters 3 and 4 follows our paper [19] closely. We set out some of the notation that we use and define variables such as the average across the population. The cloning algorithm itself employs a two-step approach of simulating dynamics and cloning systems such that those that exhibit the relevant rare events proliferate.

The algorithm is capable of simulating a wide range of processes and measuring the rare events statistics of various observables. These observables can be categorised into type $\mathcal{A}$ observables and type $\mathcal{B}$ observables [70]. It is useful and more efficient in the case of some observables to modify the dynamics [50, 96] to assist us in focusing on the rare events of interest. Here we describe the different variables and functions associated with the cloning algorithm and how their definitions change when the dynamics are modified. Variables and quantities are set out in terms of how they are mathematically related to the processes within the algorithm and we then describe functionally how some of the processes of the algorithm are implemented computationally.

We go on to set out some definitions of other averages across the population. These averages apply to observables that are measured at a single point in time and those that are measured across a window of time. Related to these averages is a characteristic time scale that we also define. We also set out some definitions of distributions of observables. These definitions are dependent on whether or not the systems are weighted by how many descendants that they have at the final time.

The algorithm has previously been used to consider large deviations of current and activity in the SSEP [86, 113] which is described in detail in chapter 3 and in the ASEP which is a similar process [34] in which the rates in each direction are unequal and to consider large deviations in the number of events in the contact process. Additionally, it has been used to study kinetically constrained models of glass formers. In particular, the Fredrickson-Andersen model, the East model and constrained lattice gas models [48, 47]. It has also been used to study nonlinear dynamical systems [112]. Other results for large deviations in the SSEP [37], ASEP [36, 42, 38, 24] and the East model [109] have also been obtained by other methods.

There are some things that are not yet fully understood about the performance of the algorithm. One of these is how the size of the cloning interval has an effect on the accuracy of the algorithm [95]. Other properties of the performance of the algorithm are understood. For example, in their two-part paper [95, 64], Nemoto, Hidalgo and Lecomte have investigated how an estimator of the large deviation function converges towards its true value. They have found that as the scalings of the estimator obey a power law they present no characteristic values above which corrections to the value obtained from the estimator become negligible. Hidalgo [62] has gone on to investigate how the validities of these scalings change for large system sizes.

2.1 Large Deviation Theory for Time Averaged-Quantities

Consider a physical system with some stochastic dynamics. The state of the system at time t is $\mathcal{C}_t$, and let A_t be a (random) observable quantity that depends on the behaviour of the system during the time-interval $[0, t]$. For example, in the simple symmetric exclusion process (SSEP) [3, 85, 86], which we introduce and describe in chapter 3, one considers particles jumping on a lattice of a discrete set of sites. In this case, A_t might be the total number of particle hopping events in $[0, t]$. Alternatively, A_t might be a time integral of the form $\int_0^t b(\mathcal{C}_{t'})\mathrm{d}t'$, where b is some function that depends on the configuration. With either of these choices, one expects that A_t obeys a large deviation principle: as t gets large, the probability distribution of A_t scales as

$$\mathrm{Prob}(A_t \approx at) \sim \exp(-\pi(a)t), \tag{2.1}$$

which describes the fluctuations in A_t and means that the dominant behaviour of the probability of the observable taking a value of A_t at large values of the time

t is a decaying exponential in t and corresponds to the observable per unit time taking a value of $a = A_t/t$, see [116, 48] for details. It defines the rate function $\pi(a) = -\ln|\text{Prob}(A_t \approx at)|/t$ in the limit $t \to \infty$, for which $\pi(a) \geq 0$. We know that the rate function $\pi(a)$ is convex as we consider discrete Markov chains. This is a necessary condition for the large deviation principle to hold. Typically, there is a single $\overline{a}$ for which $\pi(\overline{a}) = 0$. Hence, one sees from the large deviation principle (2.1) that as $t \to \infty$, the distribution of $a_t = A_t/t$ concentrates on the single value $\overline{a}$, with the probability of any other value being suppressed exponentially in t. Of course, the validity of the large deviation principle (2.1) depends on the system of interest and the observable A – here we consider irreducible Markov processes with finite (discrete) state spaces, for which the large deviation principle (2.1) holds for a large set of observables A_t: for examples see [48]. There are non-Markovian systems in which large deviation principles with other powers of t are valid [58, 56, 97].

The general aims of rare-event sampling methods in this context are (i) to estimate the function $\pi(a)$, which gives the probability of the rare event; and (ii) to characterise the rare events themselves: what are the trajectories that lead to rare values of a_t? To achieve these aims, it is convenient to introduce a *biasing field* – denoted by s – which allows access to the relevant rare events. Let Θ be a trajectory of the system, during the time interval $[0, t]$, and let $P(\Theta)$ be the probability density of observing trajectory Θ, this definition is also used for example in [48]. (The distribution $P(\Theta)$ depends on the initial condition of the model. The results of the following large-deviation analysis are independent of the initial condition, but we assume for concreteness that the initial condition is taken from the steady-state probability distribution of the model, so that $P(\Theta)$ corresponds to the steady state.) Then define a new probability density function (which is sometimes referred to as a p.d.f.)

$$\tilde{P}(\Theta, s) = \frac{P[\Theta]\exp[-sA_t(\Theta)]}{Z(s,t)}, \tag{2.2}$$

where $A_t(\Theta)$ is the value of A_t associated with trajectory Θ, and

$$Z(s,t) = \langle \exp(-sA_t) \rangle, \tag{2.3}$$

is a dynamical partition function. Here and throughout, the notation $\langle\cdot\rangle$ indicates an average with respect to $P(\Theta)$. It is useful to state

$$\psi(s) = \lim_{t\to\infty} \frac{1}{t} \ln Z(s,t), \tag{2.4}$$

which is our definition of the large deviation function $\psi(s)$. (This limit certainly

exists if the LDP (2.1) holds.)

The distribution $\tilde{P}$ is parameterised by the field s. For $s = 0$ we recover the original distribution P; for $s > 0$ trajectories with large values of A_t are suppressed. Suppose that $\mathcal{O}$ is an observable whose value in trajectory Θ is $\mathcal{O}(\Theta)$. Then its average value with respect to $\tilde{P}$ can be obtained as

$$\langle \mathcal{O} \rangle_s = \int \mathcal{O}(\Theta) \tilde{P}(\Theta, s) \mathrm{d}\Theta = \frac{\langle \mathcal{O} \exp(-sA_t) \rangle}{\langle \exp(-sA_t) \rangle}. \tag{2.5}$$

The advantage of introducing the field s is that averages of the form $\langle \mathcal{O} \rangle_s$ can often be evaluated by some numerical or analytical method. The average of the observable of interest A_t can be computed by calculating the scaled cumulant generating function $\psi(s)$ and then calculating its derivative with respective to s. In the absence of dynamical phase transitions, one may then obtain the rate function $\pi(a)$ in the large deviation principle (2.1) as

$$\pi(a) = \max_s [-\psi(s) - sa], \tag{2.6}$$

the Legendre transform of the large deviation function $\psi(s)$ which we have derived in Appendix A. Moreover, if the value of s that achieves this maximum is s_a^* then trajectories obtained from the distribution $\tilde{P}(\Theta, s_a^*)$ are representative trajectories associated with the rare event $a_t = a$ discussed above in the large time limit [27]. In this case, the numerical results can then achieve both the aims (i) and (ii) given above. The situation in the presence of dynamical phase transitions will be discussed below (chapter 3).

Note that the function ψ is a scaled cumulant generating function [48]: one has

$$\lim_{t \to \infty} \langle A_t / t \rangle_s = -\psi'(s),$$

where the prime denotes a derivative. One also has an associated susceptibility (scaled variance) which can be obtained as

$$\lim_{t \to \infty} \frac{1}{t} \left(\left\langle A_t^2 \right\rangle_s - \langle A_t \rangle_s^2 \right) = \psi''(s).$$

2.2 Modified Dynamics

The problem with computationally simulating large deviations is that the associated events are rare [113]. The trajectories with the largest values of $P(\Theta) \exp[-sA_t(\Theta)]$ that dominate $\langle e^{-sA_t} \rangle$ are not necessarily those with the largest values of $P(\Theta)$ and

hence may not dominate the population of trajectories under the natural dynamics. To assure that the relevant trajectories are sufficiently sampled it is often useful to modify the dynamics [50, 96].

It is important for us to define two types of observable that we may measure the large deviations of. Firstly type $\mathcal{A}$ observables [70, 71] that change in value every time that the systems change configuration. These can be written in the form

$$\mathcal{A}_t = \sum_{k=0}^{K-1} \alpha(\mathcal{C}_k, \mathcal{C}_{k+1}), \tag{2.7}$$

where $\alpha(\mathcal{C}_k, \mathcal{C}_{k+1})$ is the change in the value of the observable between configurations $\mathcal{C}_k$ and $\mathcal{C}_{k+1}$. Examples of type $\mathcal{A}$ observables are the activity where $\alpha(\mathcal{C}_k, \mathcal{C}_{k+1}) = 1$ and the current where $\alpha(\mathcal{C}_k, \mathcal{C}_{k+1}) = \pm 1$ depending on the direction that a particle moves. The second type of observable that we define are type $\mathcal{B}$ observables [70, 71], which are time-integrated quantities of the form

$$\mathcal{B}_t = \int_0^t dt' B(t'), \tag{2.8}$$

where $B(t)$ only depends on the configuration of the system at time t. Observables that can be taken as type $\mathcal{B}$ observables include the escape rate and Fourier components of the density. Other observables such as three-point quantities have large deviation principles. Three-point quantities are defined as $G_t = \sum_{k=0}^{K-2} \gamma(C_k, C_{k+1}, C_{k+2})$ and are dependent on the system's configuration at three different points in time.

Under natural dynamics $P(\Theta)$ is the path probability density function of trajectories. When we modify the dynamics of the process the trajectories follow a path p.d.f. denoted by $\hat{P}(\Theta)$. Both of these path probability density functions are normalized. We can relate these two p.d.f.'s

$$P(\Theta) = \hat{P}(\Theta) \exp(-\hat{U}_t(\Theta)), \tag{2.9}$$

where $\hat{U}_t(\Theta)$ is a weight function that depends on a trajectory Θ. The way in which dynamics are modified depends on the process [86, 48, 63]. We give an example of how we modify dynamics in section 3.1 which in practice means modifying the rates at which the process transitions between configurations. The p.d.f. $\tilde{P}_t(\Theta, s)$ can then be obtained using

$$\tilde{P}_t(\Theta, s) = \frac{1}{Z(s,t)} \hat{P}(\Theta) \exp(-[\hat{U}_t(\Theta) + sA_t(\Theta)]), \tag{2.10}$$

from the modified dynamics and is normalized by the partition function $Z(s,t)$. Alternatively the p.d.f. $\tilde{P}_t(\Theta, s)$ may still be obtained from the p.d.f. $P(\Theta)$ under the natural dynamics as in equation (2.2). The weight

$$\Upsilon_t(\Theta) = \exp[-\hat{U}_t(\Theta) - sA_t(\Theta)], \tag{2.11}$$

should therefore be associated with each trajectory to obtain the p.d.f. $\tilde{P}(\Theta)$ under the modified dynamics. This can be done by importance sampling [51] where we clone trajectories so that the number of copies of each system is proportional to the weight Υ_t. As the integral of $\tilde{P}(\Theta)$ across the trajectory space must be equal to 1, we obtain that

$$Z(s,t) = \int \Upsilon_t(\Theta)\hat{P}(\Theta)d\Theta, \tag{2.12}$$

so that the partition function $Z(s,t)$ is the average of the cloning factor $\Upsilon_t(\Theta)$ across the trajectory space when each trajectory is weighted by the probability density function of the modified dynamics. When we calculate type $\mathcal{A}$ and type $\mathcal{B}$ observables there are examples where we do not modify the dynamics. In this situation an integral of equation (2.2) gives us

$$Z(s,t) = \int \Phi_t(\Theta)P(\Theta)d\Theta, \tag{2.13}$$

where

$$\Phi_t(\Theta) = \exp[-sA_t(\Theta)], \tag{2.14}$$

is the weight that should be associated with trajectories under the natural dynamics to obtain the probability density distribution $\tilde{P}(\Theta)$. This can also be done by importance sampling where each trajectory is copied a number of times proportional to its associated weight Φ_t.

2.3 Cloning Algorithm

The rate function $\pi(a)$ is difficult to measure at extreme values of a so the algorithm runs at a fixed value of a bias parameter s which fixes the average of a at some value. We can then make a direct calculation of the large deviation function $\psi(s)$ at this fixed value of s. By calculating $\psi(s)$ at a range of values of s we can compute the rate function from equation (2.6), $\pi(a) = \max_s[-\psi(s) - sa]$ if $\pi(a)$ is convex [48]. We know that the rate functions that we consider are convex as we are considering finite discrete Markov chains. We can hence use the large deviation principle (2.1) to compute the probability function $\text{Prob}(A_t \approx at)$ at extreme values of $a = A/t$.

This section gives an overview of the cloning algorithm [50, 86], further details are provided in chapter 4 below. The cloning algorithm is a method that draws on earlier work by Grassberger [51] and Diffusion Quantum Monte Carlo methods [2, 103]. It is one of the two dominant computational methods for computing the large deviations of time-averaged quantities alongside transition path sampling [16].

One considers a population of n_c copies (or clones) of the system, evolving over a total observation time t_{obs} where the field s is a fixed parameter. To obtain accurate estimates of $\psi(s)$ and averages such as $\langle \mathcal{O} \rangle_s$, one requires a limit of large n_c and t_{obs}. The dependence of the results of the algorithm on n_c and t_{obs} will be discussed in section 4.3 below.

The total time t_{obs} is split into intervals of length Δt, so the number of such intervals is $M = t_{obs}/\Delta t$. Within each step of the algorithm, each clone evolves independently for a time Δt. An importance sampling method where some clones are deleted and others copied in order to bias the system towards the rare events of interest is described in [51]. In our implementation we perform a form of importance sampling at the end of the interval. In the importance sampling stage, the population size is held strictly constant. Modified algorithms with variable populations are also possible. These two sub-steps – of independent stochastic evolution followed by importance sampling – are each repeated M times. The parameter Δt (or equivalently M) can be chosen according to the problem of interest and the method can (in principle) give exact results for any value of Δt. In practice, Δt has significant effects on the accuracy of the results obtained: this is discussed in section 4.2 below.

The key feature of the method is that for any trajectory Θ, one may write

$$A_{t_{obs}}(\Theta) = \sum_{\beta=1}^{M} A^{\beta}(\Theta), \tag{2.15}$$

where $A^{\beta}(\Theta)$ is the contribution to $A_{t_{obs}}(\Theta)$ from the time interval $[t_{\beta-1}, t_{\beta}]$, in which $t_{\beta} = \beta \Delta t$. The cloning algorithm and a lot of the theory that surrounds it relies on the fact that A_t can be decomposed in this way. Using definition (2.14) and letting the trajectory of the ith clone be Θ_i, one may compute

$$\Phi_{i\beta} = \exp(-sA^{i\beta}), \tag{2.16}$$

for clone i in time interval β where $A^{i\beta} = A^{\beta}(\Theta_i)$ is the value of A^{β} in the trajectory of the ith clone. Within the importance sampling step of the algorithm, the (average) number of descendants of clone i is proportional to $\Phi_{i\beta}$. In the cloning language, one may think of $\Phi_{i\beta}$ as an evolutionary fitness.

The definition of $\psi(s)$ is $\frac{1}{t}\ln(\langle\Phi_t\rangle)$ in the limit $t \to \infty$. Moreover, the estimate of $\psi(s)$ provided by the algorithm is

$$\hat{\psi}_{n_c,t_{obs}}(s) = \frac{1}{t_{obs}} \sum_{\beta=1}^{M} \ln\left(\frac{1}{n_c}\sum_{i=1}^{n_c}\Phi_{i\beta}\right), \tag{2.17}$$

and one has $\lim_{n_c,t_{obs}\to\infty}\hat{\psi}_{n_c,t_{obs}}(s) \to \psi(s)$ [95]. This estimator becomes exact as $n_c, t_{obs} \to \infty$ because its systematic and statistical errors vanish. As discussed in section 2.2 we are free to modify the dynamics of the process to focus on the most relevant trajectories and obtain the partition function from equation (2.12) which is the average of weights $\Upsilon_{i\beta}$ under the modified process as opposed to the average of $\Phi_{i\beta}$ under the natural process. Hence

$$\tilde{\psi}_{n_c,t_{obs}}(s) = \frac{1}{t_{obs}} \sum_{\beta=1}^{M} \ln\left(\frac{1}{n_c}\sum_{i=1}^{n_c}\Upsilon_{i\beta}\right), \tag{2.18}$$

is an estimate of $\psi(s)$ across the modified process because $\lim_{n_c,t_{obs}\to\infty}\tilde{\psi}_{n_c,t_{obs}}(s) \to \psi(s)$ [95]. As previously, the estimator becomes exact as $n_c, t_{obs} \to \infty$ because the systematic and statistical errors that the estimator is subject to vanish. Clearly the values of $\Upsilon_{i\beta}$ depend on how the dynamics of each trajectory are modified. The total modification of the trajectory depends on a weight $\hat{U}_{t_{obs}}(\Theta)$ defined in definition (2.9) which can be expressed as the sum of contributions

$$\hat{U}_{t_{obs}}(\Theta) = \sum_{\beta=1}^{M} \hat{U}^{\beta}(\Theta), \tag{2.19}$$

from each cloning interval. We can then compute

$$\Upsilon_{i\beta} = \exp(-\hat{U}^{i\beta} - sA^{i\beta}), \tag{2.20}$$

where under the importance sampling step of the algorithm, the (average) number of descendants of clone i is proportional to $\Upsilon_{i\beta}$. In cloning language, one may think of $\Upsilon_{i\beta}$ as an evolutionary fitness under the modified dynamics.

2.4 Computational Procedures

The algorithm generates a value of the large deviation function $\psi(s)$ by computing the partition function $Z(s, t_{obs}) \equiv \langle\exp(-sA_{t_{obs}})\rangle$, where this expression defines the partition function. As discussed in section 2.3, the dynamics are run for one cloning

interval at a time and each has an index β. On each clone the dynamics run from the start of the cloning interval at time $t' = t_\beta$ to the end of the cloning interval at time $t' = t_\beta + \Delta t = t_{\beta+1}$. When the natural dynamics are simulated, the cloning factor $\Phi_{i\beta}$ associated with each clone i is calculated and the value of the estimator of the large deviation function $\hat{\psi}$ defined in equation (2.17) updated

$$\hat{\psi}_{n_\mathrm{c},t_\beta+\Delta t} = \frac{t_\beta \hat{\psi}_{n_\mathrm{c},t_\beta} + \ln\left(\Phi_T/n_\mathrm{c}\right)}{t_\beta + \Delta t}, \tag{2.21}$$

where the variable

$$\Phi_T = \sum_{i=1}^{n_\mathrm{c}} \Phi_{i\beta}, \tag{2.22}$$

is the total of the cloning weights across all of the systems. When we have run the modified dynamics from the start to the end of a cloning interval we calculate the cloning factor $\Upsilon_{i\beta}$ associated with each system. We then update the value of the estimator $\tilde{\psi}$ defined in equation (2.18)

$$\tilde{\psi}_{n_\mathrm{c},t_\beta+\Delta t} = \frac{t_\beta \tilde{\psi}_{n_\mathrm{c},t_\beta} + \ln\left(\Upsilon_T/n_\mathrm{c}\right)}{t_\beta + \Delta t}, \tag{2.23}$$

where the variable

$$\Upsilon_T = \sum_{i=1}^{n_\mathrm{c}} \Upsilon_{i\beta}, \tag{2.24}$$

is the total of the cloning weights across all of the systems. For both types of observable and regardless of whether the dynamics are modified $Z(s,0) = 1$ at the beginning of the simulation.

After the value of the partition function has been updated the importance sampling takes place. In this stage of the algorithm, we clone all of the properties associated with a system that are required to be copied, these are sometimes referred to as components. In chapters 3, 4 and 5 we analyse several observables associated with the algorithm such as the estimator for the observable we are biasing, the Fourier components of density $\delta\rho_q$ and the activity per cloning interval K^β. These require the trajectories to be obtained by extrapolation backwards through time, this is achieved by these observables being cloned along with the properties of the system during the cloning stage [113].

For some processes and observables the algorithm requires a large number of systems n_c and units of time t_obs to obtain results, this may lead to the code having long run times. Researchers that use the cloning algorithm may need to attain results at high speeds or efficiencies and so in chapter 6 we investigate different computa-

tional implementations of the algorithm in detail. To improve the observed speeds and efficiencies we go on to use computational parallelism, employing OpenMP, MPI and OpenMP-MPI hybridisation.

2.5 Estimating Averages with Respect to $\tilde{P}$

To estimate averages of the form $\langle \mathcal{O} \rangle_s$, as defined in definition 2.5, we consider the entire population at the final time t_{obs}. Each system has a trajectory associated with it that is generated by following that history of the system backwards in time to the initial time $t = 0$. Many members of the final population will have common ancestors which means that up to some time $t < t_{\text{obs}}$ their trajectories are identical. Therefore not all trajectories are independent samples from the p.d.f. $\tilde{P}(\Theta)$. If the history of the ith clone corresponds to a trajectory $\hat{\Theta}_i$ one may then estimate the general expectation value,

$$\langle \mathcal{O} \rangle_s \approx \frac{1}{n_{\text{c}}} \sum_{i=1}^{n_{\text{c}}} \mathcal{O}(\hat{\Theta}_i), \tag{2.25}$$

of an observable $\mathcal{O}$ as defined in equation (2.5). The value $\mathcal{O}(\Theta)$ is the value of observable $\mathcal{O}$ in trajectory Θ. The equality becomes exact [95] as $n_{\text{c}}, t_{\text{obs}} \to \infty$.

The observable $\mathcal{O}$ may depend on any aspect of the trajectory Θ. Let F_t be a function that depends on the state of the system at time t, such as a Fourier component density or the escape rate. We can also generalise the function and definitions below to include observables such as the activity per cloning interval K^β and cloning factor Υ^β that depend on the trajectory in a preceding finite time window $[t - \Delta t, t]$. At no bias $s = 0$, the probability density function P is time translation invariant (TTI) and does not depend on t. For all other values of s the average $\langle F_t \rangle_s$ is time-dependent. The average exhibits transient regimes [70] in which its value changes. These transient regimes exist at small t close to the initial time and at small $t_{\text{obs}} - t$ close to the final time. There is a time-translation invariant regime between the transient regimes in which the averages do not change with time. We can say that

$$\langle F_t \rangle_s = \begin{cases} F_{\text{i}}(t), & t \lesssim \tau \\ F_{\text{f}}(t_{\text{obs}} - t), & (t_{\text{obs}} - t) \lesssim \tau \\ F_\infty, & \text{otherwise,} \end{cases} \tag{2.26}$$

where τ is a characteristic time scale that describes the length of the transient regimes. The functions $F_{\text{i}}(t)$, $F_{\text{f}}(t_{\text{obs}} - t)$ describe the average in the transient

regimes at small t and small $t_{\text{obs}} - t$ [94]. The asymptotic value F_∞ describes the time translation invariant value of the variable F_t between the transient regimes and so is not a function of time. We measure the probability distributions of F_t and define these as

$$p_{t_{\text{obs}}}(F, t) = \langle \delta(F - F_t) \rangle_s, \tag{2.27}$$

so that this is the probability density to observe the value F of the observable F_t at time t in trajectories of length t_{obs} with distribution $\tilde{P}$. To characterise the TTI regime we choose α between 0 and 1 so that $\alpha t_{\text{obs}} \gg \tau$ and $(t_{\text{obs}} - \alpha t_{\text{obs}}) \gg \tau$ when $t_{\text{obs}} \to \infty$ because $\alpha t_{\text{obs}} \to \infty$ and $(t_{\text{obs}} - \alpha t_{\text{obs}}) \to \infty$. We then define

$$p_{\text{ave}}(F) = \lim_{t_{\text{obs}} \to \infty} p_{t_{\text{obs}}}(F, \alpha t_{\text{obs}}), \tag{2.28}$$

where we know αt_{obs} must be in the TTI regime independent of α. We can use this to state that the time translation invariant value $F_\infty = \int F p_{\text{ave}}(F) dF$. We also define another distribution

$$p_{\text{end}}(F) = \lim_{t_{\text{obs}} \to \infty} p_{t_{\text{obs}}}(F, t_{\text{obs}}), \tag{2.29}$$

which is the distribution of F at the final time $t = t_{\text{obs}}$. After the importance sampling step of the cloning algorithm the clone population is distributed as p_{end}. The p_{ave} distribution, however, represents the clone population obtained by constructing the trajectories back through time. When we pause the simulation at $t = t_{\text{obs}}/2$ and store the observables of the trajectories they are distributed by p_{end}. When we continue the simulation to t_{obs} and weight each of these stored observables by the number of descendants of its associated trajectory that exist at the final time they are distributed by p_{ave}. There is a detailed discussion of this in [94].

Chapter 3

Dynamical Phase Transition in the SSEP

In this chapter we consider the Simple Symmetric Exclusion Process (SSEP) [87, 111, 80, 79] a simple lattice model. The SSEP is used widely for modelling dynamical systems in the fields of physics, chemistry and biology [44] and in theories of stochastic processes.

The large deviations of activity in this process have previously been studied [85, 3]. In the study of large deviations it has been observed that simple systems exhibit behaviours such as intrinsically dynamic phase phase transitions [8, 9, 12, 13, 47, 48]. This is also the case in the SSEP where a phase transition exists in the properties of the system [85]. This phase transition exists as $L \to \infty$. Under a scaling of the bias s the position of the phase transition is consistent with the large deviations of all large systems. The value of the large deviation function and its cumulants are known up to this phase transition for finite-L. In the large-L limit it is also understood how the large deviation function and hence its derivatives behave up to and beyond the phase transition.

Our contribution to the understanding of the large deviations in activity in the SSEP is to obtain the large deviation function from the cloning algorithm set out by Giardinà, Peliti and Kurchan [50] and by Lecomte and Tailleur [86] based on ideas from Grassberger [51] at finite values of L above the phase transition. We also obtain measures of the clustering of particles as the bias s is varied. Specifically we present typical trajectories at values of the scaled bias around the phase transition and obtain values of the first component of Fourier density as a function of the scaled bias. The magnitude of this variable increases as the particles become more clustered.

3.1 Model and Choice of Dynamical Observable

We consider a one-dimensional lattice of L sites with periodic boundaries. The lattice is occupied by N particles and each site can be occupied by at most one particle. In the symmetric simple exclusion process (SSEP) step sizes Δt_h between attempted hops are drawn from a probability density function,

$$P(\Delta t_h) = 2N \exp(-2N\Delta t_h), \tag{3.1}$$

under the natural dynamics because this gives an average time of $\Delta t_h = (2N)^{-1}$ between attempted hops. At each time step a random particle is picked and a hop attempted left or right with equal probability. This means that each particle attempts to hop on average with rate 1 in each direction as in figure 3.1. The attempted hop is successful if the neighbouring site is unoccupied. There are other techniques that may make the basic dynamics more efficient such as the Gillespie algorithm [78]. We deliberately did not optimise the method by which we simulate the dynamics of individual systems as we are interested in the performance of the cloning algorithm when there is some fixed method for simulating individual systems. Let the occupancy of site i be n_i. The model obeys detailed balance. In its equilibrium (steady) state, the occupancy of each site is independent: $n_i = 1$ with probability $\rho = N/L$ and $n_i = 0$ with probability $1 - \rho$.

The observable A_t considered here is the total number of (successful) hops in the time-interval $[0, t]$. This is the same as the number of configuration changes and is hence a type $\mathcal{A}$ observable. We denote this observable K_t and refer to it as the activity. Within the steady state of the model, one has $\langle K_t/t \rangle = 2L\rho(1-\rho)$, since the rate of attempted hops is $2N = 2\rho L$ and the expected fraction of successful hops is equal to the probability $(1 - \rho)$ that the destination site is unoccupied.

It is notable that the probability density function $\tilde{P}(\Theta, s)$ may be written in a number of alternative ways. Consider a general stochastic model with configurations $\mathcal{C}$, transition rates $W(\mathcal{C} \to \mathcal{C}')$, and define the escape rate $r(\mathcal{C}) = \sum_{\mathcal{C}'} W(\mathcal{C} \to \mathcal{C}')$. Taking $A_t = K$, one has

$$\tilde{P}(\Theta, s) = \frac{1}{Z(s,t)} \left[\prod_{i=0}^{K-1} W(\mathcal{C}_i \to \mathcal{C}_{i+1}) \mathrm{e}^{-s} \exp\Big(- r(\mathcal{C}_i)(t_{i+1} - t_i) \Big) \right]$$

where $t_0 = 0$ and t_i is the time at which hop i takes place for $i = 1 \ldots K$. Given that activity is a type $\mathcal{A}$ observable we are free to modify the dynamics to focus on the relevant region of the trajectory space. Consider a modified model in which all rates are scaled by a factor of e^{-s}, so that $W_s(\mathcal{C} \to \mathcal{C}') = \mathrm{e}^{-s} W(\mathcal{C} \to \mathcal{C}')$ and the

scaled escape rates are $r_s(C) = \mathrm{e}^{-s} r(C)$ so

$$\tilde{P}(\Theta, s) = \frac{1}{Z(s,t)} \left[\Upsilon(\Theta) \prod_{i=0}^{K-1} W_s(\mathcal{C}_i \to \mathcal{C}_{i+1}) \exp\Big(- r_s(\mathcal{C}_i)(t_{i+1} - t_i) \Big) \right],$$

where we have introduced a modified reweighting parameter $\Upsilon(\Theta) = \exp(\int_0^t [r_s(\mathcal{C}) - r(\mathcal{C})]dt)$ associated with each trajectory. This means that the trajectory ensemble for the original model and a reweighting parameter $\mathrm{e}^{-sA_t(\Theta)}$ is the same as the ensemble obtained from the modified model with a modified reweighting parameter $\Upsilon(\Theta)$. This can be stated as $\tilde{P}(\Theta, s) = \Upsilon \hat{P}(\Theta)/Z(s,t)$ where $\hat{P}(\Theta)$ is the probability density of following a trajectory under the modified weights. Under the modified dynamics, step sizes $\Delta\tilde{t}_h$ are drawn from a probability density function

$$P(\Delta\tilde{t}_h) = 2Ne^{-s} \exp(-2Ne^{-s}\Delta\tilde{t}_h), \tag{3.2}$$

as opposed to the $P(\Delta t_h)$ probability density function described in equation (3.1) because under equation (3.2) they have an average time of $\Delta\tilde{t}_h = (2Ne^{-s})^{-1}$ between attempted hops. As under the natural dynamics we choose a random particle at each time step and attempt to hop left or right with equal probability. Particles then attempt to hop on average with rate e^{-s} in each direction. We show in Appendix B that when the time steps between attempted hops are drawn from equation (3.2) that in configuration $\mathcal{C}$ successful hops occur at the modified escape rate $r_s(\mathcal{C})$. In Appendix A we state a master equation to describe the evolution of the path p.d.f. $\hat{P}(\Theta)$ under the SSEP with modified dynamics and in Appendix C we show that it is satisfied by the hops occurring at the modifed escape rate $r_s(\mathcal{C})$.

By integrating over all trajectories we find that

$$Z(s,t) = \int \Upsilon(\Theta)\hat{P}(\Theta)d\Theta \approx \sum_{i=1}^{n_c} \Upsilon_i / n_c,$$

where Υ_i is the cloning factor associated with the trajectory of the ith clone. This construction is a general one: one is free to modify the transition rates of the model, as long as one makes a corresponding modification to the reweighting factor. This fact was exploited in [94], to aid computational efficiency. In Appendix D we derive the values of the cloning factors for the modification of the SSEP that we have made and in Appendix A derive a master equation to describe the evolution of $\tilde{P}(\Theta)$.

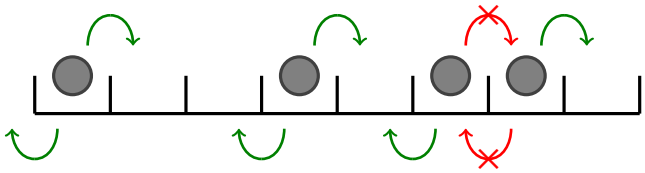


Figure 3.1: An example of SSEP on a one-dimensional lattice of 8 sites with periodic boundary conditions and a density $\rho = 0.5$. Each hop is attempted with rate 1.

3.2 Theoretical Analysis of Dynamical Phase Transition

There is a well-understood phase transition [8, 9, 12, 13, 85] that occurs in the SSEP, when considering large deviations of the activity K_t. For systems of finite size, the large deviation function $\psi(s)$ and rate function $\pi(a)$ are analytic and there is no phase transition. However, it is interesting to fix the density $\rho = N/L$ and take the conventional thermodynamic limit $L \to \infty$ and the infinite time limit $t \to \infty$. One then expects K_t to be proportional to the system size, so it is useful to define the activity per site per unit time $k(s) = \lim_{L,t\to\infty} \langle K_t \rangle_s / (Lt)$. It is important that we take $t \to \infty$ before $L \to \infty$. For the cases where we know, these limits commute.

Phase transitions correspond to singularities in $k(s)$. In this case we have $k(0) = 2\rho(1-\rho)$ but $k(s) = 0$ for all $s > 0$ [47], so there is a discontinuity at $s = 0$, corresponding to a first-order phase transition. It is known [47] that there is a lower bound on $\psi(s) \geq -\min_{\mathcal{C}} r(\mathcal{C})$. This bound is derived from the fact the large deviation function $\psi(s)$ is equal to the largest eigenvalue of a transition matrix that represents that process. The lower bound on its value is determined using a variational principle that is valid for any symmetric operator. In any system where $L \to \infty$ one can find configurations such that the subextensive escape rate $r/L \to 0$. In the case of the SSEP on a one-dimensional lattice, when all particles are in one cluster $r = 2$ independent of L and $r/L \to 0$ as $L \to \infty$. Given the lower bound on $\psi(s)$, this means that $\psi(s)/L \geq 0$ when $L \to \infty$.

The observation of a phase transition was made by Bodineau and Derrida [12]. They have found that macroscopic fluctuation theory predicts correctly the large deviation function. Macroscopic fluctuation theory is based on the probability of fluctuations in thermodynamic variables and currents and provides a macroscopic treatment of stationary nonequilibrium states for driven diffusive systems [10]. The numerical results that Bodineau and Derrida have obtained are consistent with the second order phase transition that they predicted to exist when measuring large

deviations in the current. They predicted that these results could be confirmed by solving Bethe Ansatz equations. The phase transition that they have observed is similar to the phase transition that we observe in the activity in that it is between a regime in which the density remains constant and one in which the density becomes time dependent.

In the approach of Appert-Rolland et al [3], the values of the activity at finite values of L below the phase transition during a long period of time t_{obs} are obtained. They have also obtained values of the first cumulants of the activity K. For large system sizes these cumulants take universal scaling forms as does the large deviation function. They show how these scaling forms can be calculated for SSEP using a Bethe ansatz method. The scaling forms can be extracted from a detailed analysis of finite size effects similar to what has been developed for quantum spin chains in the context of string theory [7, 53].

Appert-Rolland et al have also derived the existence of a phase transition in the large deviation function $\psi(s)$. They have obtained an expression for $\psi(s)$ by stating it as an expansion of the values of its cumulants that they have obtained. This expression is only valid in the limits $s \to 0$ and $L \to \infty$. As the bias s tends towards a critical value, the expansion becomes singular. This divergence is what implies the existence of a phase transition.

When we consider values of the bias s above the phase transition, we compare our results to the results of Lecomte, Garrahan and van Wijland [85]. They observed that the phase transition is from a homogeneous state to a kink-like profile. This occurs as the trajectories' activity is lowered to a below average value. They have obtained values of the large deviation function in the limit $L \to \infty$ at values of the bias above the phase transition. They do this by finding solutions to elliptical equations in the inactive regime which they obtain from macroscopic fluctuation theory.

3.3 Properties of Dynamical Phase Transition

The phase transition can physically be seen in figure 3.2 where there is a transition from a homogeneous state (where particles are distributed evenly throughout the system) to an inhomogeneous ("phase-separated") state in which the particles are segregated into a dense and a dilute region.

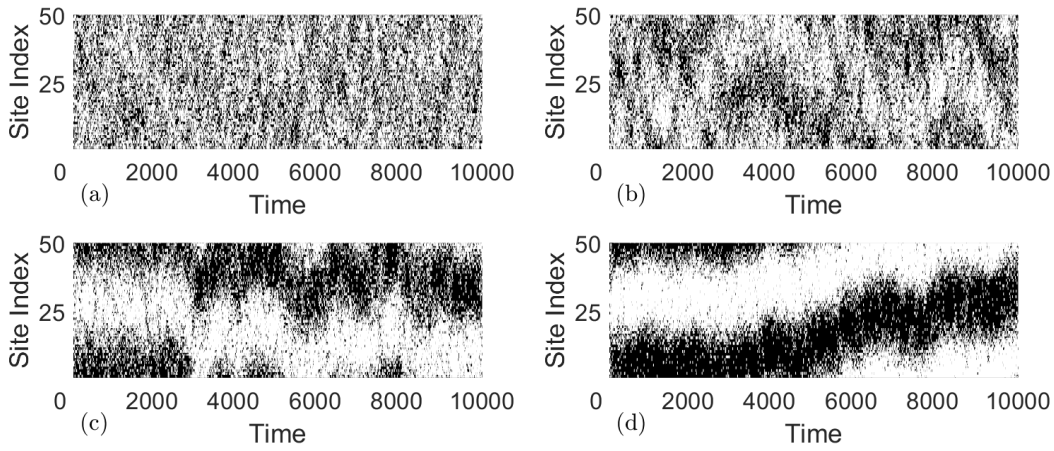


Figure 3.2: Sample trajectories of SSEP with $L = 50$, $\rho = 0.5$ and $t_{\text{obs}} = 10^4, n_{\text{c}} = 10^5$. (a) $s = 0$, the equilibrium state; (b) $s = 0.008$, showing evidence of transient clusters; (c) $s = 0.012$, in which a single large cluster has formed; (d) $s = 0.020$, with most of the particles in a well-defined single cluster. The corresponding values of λ as defined in definition 3.4 are $0, 20, 30, 50$ and the critical value of λ is $\lambda_c = 2\pi^2 \approx 19.7$.

The clustering of the particles that occurs as the bias s is increased is related to other observables such as the Fourier components of the density which we define as

$$\delta\rho_n = \frac{1}{\sqrt{L}} \sum_{j=1}^{N} \exp(-2\pi \mathrm{i} n X_j / L), \tag{3.3}$$

where X_j is the index of the site occupied by particle j and $n = 0, 1, \ldots L-1$. We obtain results for the first Fourier component of the density which corresponds to $n = 1$. We see in figure 3.3 that above the phase transition this Fourier component of the density grows as the particles cluster. We also see above the phase transition $\lambda > \lambda_c$ that for each value of λ the values of $\langle |\delta\rho_1|^2 \rangle_s$ are approximately proportional to L.

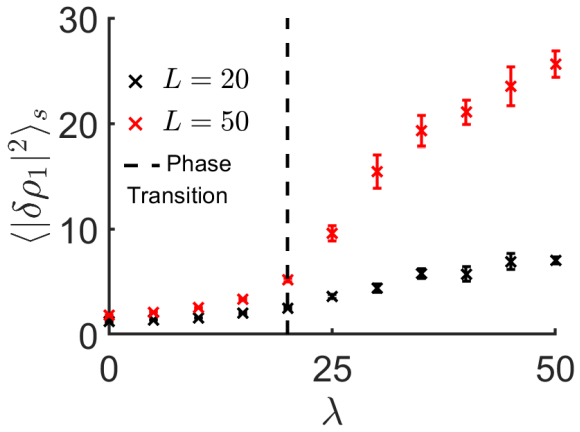


Figure 3.3: The mean square value of the first Fourier component of the density, $\langle|\delta\rho_1|^2\rangle_s$, measured at $t = t_{\text{obs}}/2$ with $t_{\text{obs}} = 10^4, n_c = 10^5$, for various L, λ as defined in definition 3.4 at $\rho = 0.5$. The phase transition occurs at $\lambda_c = 2\pi^2$: for $\lambda > \lambda_c$ one expects the system to become inhomogeneous so that $\langle|\delta\rho_1|^2\rangle_s \propto L$, consistent with the data.

To investigate in detail, we make the same scaling as Lecomte et al [85] and let

$$\lambda = sL^2 \tag{3.4}$$

and consider the limit of large L at fixed λ. This is also the same scaling as Appert-Rolland et al [3] to order L. This allows us to zoom in on the jump in k and resolve it as a (continuous) crossover from large k to small k, whose width is of order L^{-2}. This is analogous to finite-size scaling in equilibrium systems. The crossover is described by the function

$$\mathcal{K}(\lambda) = k(\lambda/L^2), \tag{3.5}$$

which may be computed using macroscopic fluctuation theory [85]. For $\lambda < 2\pi^2 (\approx 20)$ one has $\mathcal{K}(\lambda) = 2\rho(1-\rho)$, independent of λ. For $\lambda > 2\pi^2$, the function $\mathcal{K}(\lambda)$ decreases smoothly from $2\rho(1-\rho)$ to zero. The derivative $\mathcal{K}'(\lambda)$ is discontinuous at $\lambda = 2\pi^2$. When viewed on this scale, the transition has some features of a continuous phase transition [22, 4].

Note this situation is different from classical finite-size scaling and from other first-order dynamical phase transitions [96]. The reason is that we have taken $t \to \infty$ before large L so we can think of a system on a long cylinder. In that case one expects a first-order phase transition to happen by the formation of large domains that wrap around the cylinder, with domain walls perpendicular to the time axis. This leads to a natural analogy with thermodynamic phase transitions in cylindrical geometries. However, since the number of particles in the SSEP is equal at every

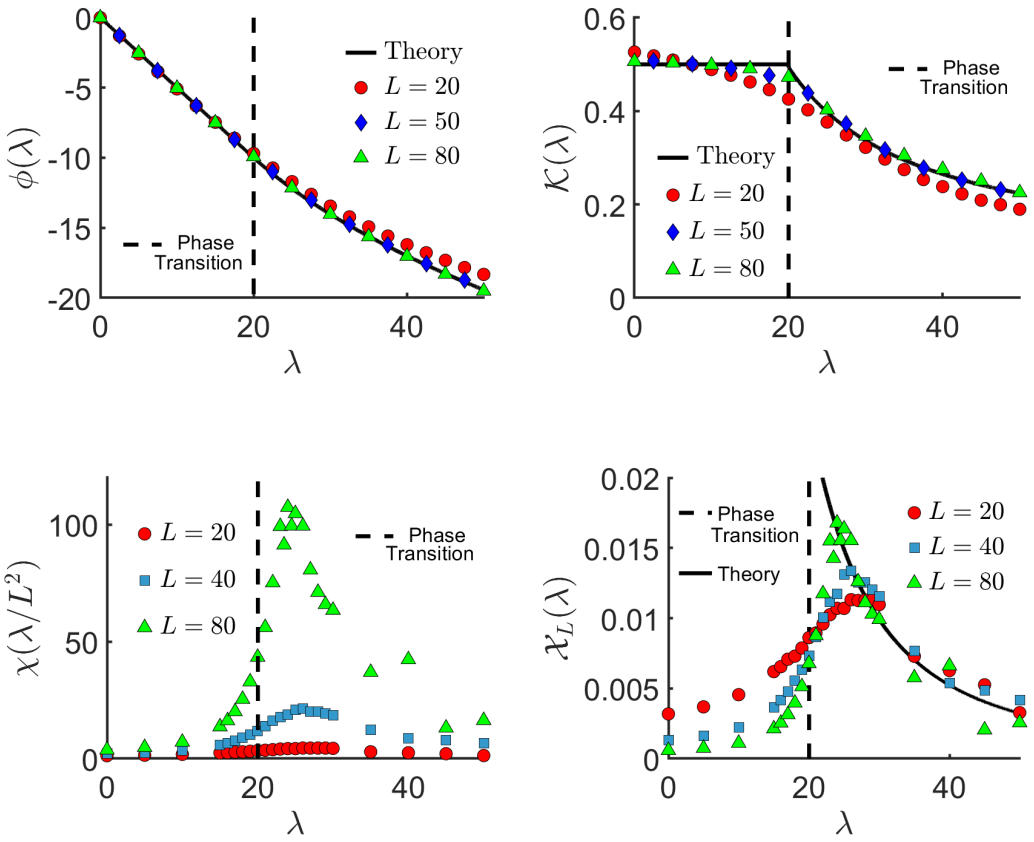


Figure 3.4: Results obtained from our implementation of the cloning algorithm. Results are obtained with $t_{\text{obs}} = 10^4$ for all system sizes, 10^5 clones for $L = 20, 40, 50$ and 10^6 clones for $L = 80$. The density is $\rho = 0.5$, the cloning interval is $\Delta t = 10$ and the results are averaged across 10 independent realizations. The vertical dashed line shows the position of the phase transition ($\lambda_c = 2\pi^2$). The theoretical values are obtained by solving Euler-Lagrange equations as set out in [85] and taking derivatives using finite differences. The theoretical values are valid in the limit $L \to \infty$.

time, the domain wall between dense and dilute phases must run *parallel* to the time axis. This leads to the unusual finite-size scaling behaviour described here.

It is useful to rescale the dynamical free energy

$$\phi(\lambda) = \lim_{L\to\infty} L\psi(\lambda/L^2), \tag{3.6}$$

so that $\mathcal{K}(\lambda) = -\phi'(\lambda)$. This is the approach used by Lecomte, Garrahan and van Wijland [85] and is inspired by the work of Appert-Rolland et al [3] and Bodineau and Toninelli [15]. It's also useful to define an appropriate susceptibility

$$\chi(s) = \psi''(s)L^{-1}, \tag{3.7}$$

which is a measure of the scaled variance of K_t across all of the systems. We should have that as $L \to \infty$, then $L^{-2}\chi(\lambda/L^2) \to \mathcal{K}'(\lambda)$. However, the way that this convergence happens is not clear (in particular the scaling with L of the maximal susceptibility $\chi^* = \max_s \chi(s)$). In a system that is far from any phase transition $\lim_{L\to\infty} \chi_L(s)$ should have a finite value. In figure 3.4 we see a divergence in χ which is consistent with the existence of a phase transition. We also define another measure of the susceptibility

$$\mathcal{X}_L(\lambda) = -\mathcal{K}'(\lambda) = L^{-2}\chi(\lambda/L^2), \tag{3.8}$$

which is predicted by MFT to have a finite limit as $L \to \infty$. Our numerics in figure 3.4 are consistent with this prediction but they show that measuring this limiting function requires large system sizes.

When studying properties of the large deviation function at positive values of s, Appert-Rolland et al [3] introduce a function $\sigma(\rho) = 2\rho(1-\rho)$ where $\rho = N/L$ is the density. This is a function also used by Lecomte, Garrahan and van Wijland [85]. For our purposes $\sigma(\rho) = 0.5$ as we are considering systems in which the lattice is half-full, $\rho = 0.5$. For sufficiently large systems below the phase transition it is shown in [3] that $\mathcal{K}(\lambda) = \sigma$ which they have obtained using a Bethe Ansatz method. The activity $\mathcal{K}(\lambda)$ is equal to the derivative of the large deviation function and so as stated in [85] the large deviation function takes the value $\phi(\lambda) = -\lambda\sigma$ in the limit $L \to \infty$. The measure of susceptibility $\mathcal{X}_L(\lambda)$ is equal to the derivative of the activity so we expect that $\mathcal{X}_L(\lambda) = 0$ when $0 < \lambda < 2\pi^2$ as $L \to \infty$.

Lecomte, Garrahan and van Wijland [85] have then been able to obtain theoretical values of $\phi(\lambda)$ at values of $\lambda > \lambda_c = 2\pi^2$ above the phase transition by solving Euler-Lagrange equations. These values hold in the limit $L \to \infty$. By using finite

differences to take the derivatives of $\phi(\lambda)$, we can then obtain theoretical values of the activity $\mathcal{K}(\lambda) = -\phi'(\lambda)$ and the susceptibility $\mathcal{X}_L(\lambda) = \phi''(\lambda)$. This means that above $\lambda = 0$ we have theoretical values for all three of these quantities in the large size limit.

3.4 Scaling at the Dynamical Phase Transition

We consider systems of size $L = 20$, 40, 50 and 80 at density $\rho = 0.5$ in figure 3.4. We directly measure the values of activity in the algorithm code. We also directly measure the variance in activity across the population and use this to obtain our scalings of susceptibility $\chi(s)$ and $\mathcal{X}_L(\lambda)$. In section 4.2 we investigate the optimal size of cloning interval at which to acquire results for the SSEP and find $\Delta t = 10$ to be the best in terms of speed and accuracy. In section 4.3 we measure the number of clones and units of time required for the results for the SSEP to have converged and find that $t_{\text{obs}} = 10^4$ and $n_{\text{c}} = 10^5$ ($L = 20, 50$), $n_{\text{c}} = 10^6$ ($L = 80$) are necessary. These are the parameters used to obtain the results in figure 3.4 and we use $n_{\text{c}} = 10^5$ clones for systems of size $L = 40$. These results were obtained with the [eq] method to select which systems to clone as determined in section 4.2 from the two methods set out in section 4.1 due to its superior speed and smaller errors.

The theoretical values of the activity and susceptibility are obtained by using finite differences to take the derivative of the theoretical value of the scaled large deviation function $\phi(\lambda)$. The values of $\phi(\lambda)$ are obtained using the method of Lecomte, Garrahan and van Wijland [85] of solving Euler-Lagrange equations which they have obtained from macroscopic fluctuation theory. We do not include the theoretical values obtained by Appert-Rolland et al [3] for large finite L as they only apply as $s \to 0$ and become singular as s tends towards the phase transition. For systems of size $L = 20$ we see that the algorithmic values of $\mathcal{K}(\lambda)$ and $\phi(\lambda)$ are close to but not the same as the theoretical values which are valid as $L \to \infty$. The algorithmic values align closely with the theoretical values for systems of sizes $L = 50, 80$, which suggests that $L = 50$ is enough sites for the theoretical values to be valid for most values of the bias s. When looking specifically at the activity subfigure we find that even systems of size $L = 50, 80$ are not big enough around the phase transition $\lambda_c = 2\pi^2$ for the algorithmic values and the theoretical values to agree.

One value of interest is the value of λ at which the susceptibility is maximised

$$\lambda_{\max} = \text{argmax}_\lambda \ \chi(\lambda) = \text{argmax}_\lambda \ \mathcal{X}_L(\lambda), \tag{3.9}$$

which depends on L and according to theory [85], is expected to occur at the phase transition $\lambda_c = 2\pi^2$ as $L \to \infty$. In the susceptibility subfigures in figure 3.4 we clearly see the value of $\lambda_{\max}$ vary with the system size. For each system size that we have considered the peak in susceptibility exists at a value of λ greater than the value at which the phase transition exists. In the subfigure of values of $\mathcal{X}_L$ we observe that the position of the peak decreases towards $\lambda_c = 2\pi^2$ as the system size L increases.

Another value of interest is the largest value of susceptibility

$$\kappa_{\text{eff}} = \max_{\lambda} \mathcal{X}_L(\lambda), \tag{3.10}$$

which depends on L. Lecomte, Garrahan and van Wijland [85] state that κ_{eff} takes a value of $3/4\pi^2$ in the limit of $L \to \infty$. The algorithmic results that we have obtained in figure 3.4 are clearly much smaller than this. We also observe in figure 3.4 that the value of κ_{eff} changes with the system size L. As the system size increases figure 3.4 shows that the height κ_{eff} of the peak in $\mathcal{X}_L$ increases. We observe that for each system size L the value of κ_{eff} is similar to the theoretical value of $\mathcal{X}_L(\lambda_{max})$, the theoretical value of the susceptibility at the value of λ at which we measure the maximum susceptibility for this size of system L.

One property of this region of the large deviations of the SSEP that is not understood is the rate at which the peak of the $\mathcal{X}_L(\lambda)$ increases towards the large-L limit as the number of sites L is increased for finite L. This would be an interesting way to add to the results that we have obtained. Furthermore, the value of λ at which the values of $\mathcal{X}_L(\lambda)$ crossover for finite values of L is not understood. This is the value of λ at which $\mathcal{X}_L(\lambda)$ is the same for two different system sizes. It is not clear whether it would be a similar value for all L and would be another interesting area of research in the future.

3.5 SSEP Summary

In conclusion we have obtained results that agree with the theoretical values of activity and the large deviation function below the phase transition. Above the phase transition, we have shown the similarity between our algorithmic results and the theoretical values of the large system limit $L \to \infty$. We have investigated how these results scale with L in terms of the height of the peak in susceptibility increasing and the values of the bias at which the peak in susceptibility occurs decreasing towards $s = 2\pi^2/L^2$.

We have also measured how the first Fourier component of the density evolves

from being somewhat constant below the phase transition to increasing more rapidly with λ above the phase transition. We also observe that this variable exhibits larger values as the system size is increased. Correspondingly, we find that the typical trajectories that we observe up to the phase transition exhibit clusters of particles that are unstable. Above the phase transition the clusters in these trajectories rapidly stabilise and the transition of particles between the cluster and the corresponding sparser region decreases.

Chapter 4

Algorithm Performance in the SSEP

The results obtained in chapter 3 for the SSEP are obtained at values of the bias s, λ around the phase transition and the modified SSEP dynamics are implemented using the Bortz-Kalos-Lebowitz (continuous time Monte Carlo) algorithm [17] by selecting time increments between attempted hops from equation (3.1). We require these results to be sufficiently accurate. To do this we define criteria for accuracy to measure when the algorithmic results are sufficiently close to their true value. Our results must be obtained with parameters and methods that meet these criteria. Some of these parameters affect the run time of the code which we aim to keep as low as possible. In this chapter we define two methods for selecting which systems are copied during the cloning stage of the algorithm. We test these two methods and the size of the cloning interval (Δt) in terms of how they are related to the speed and accuracy of the code. We quantify the accuracy of the code in terms of the systematic errors and statistical errors in the values of activity which are expressed by the standard deviation across several realizations.

We then measure the convergence of the algorithm with respect to two quantities: the number of units of time for which the algorithm is run t_{obs} and the size of the population n_{c}. In each case the general scaling of the estimator with respect to the quantity is known [64] but the rate of scaling and values required to meet our accuracy criteria have been obtained by us. In addition we have investigated the reasons that these quantities are required to take the values that they do and derived lower bounds on them by measuring the evolution over time and distributions of relevant observables.

4.1 Clone Selection Methods

During the cloning process systems from the current set of clones are copied to a new set of clones. Our implementation must be such that the number of times that a system is cloned is proportional to its associated $\Upsilon_{i\beta}$ value. We require that the average number of descendants of clone i approaches $n_c \Upsilon^\beta(\Theta_i)/\Upsilon_T^\beta$ where $\Upsilon_T^\beta = \sum_{i=1}^{n_c} \Upsilon^\beta(\Theta_i)$. There are a number of ways of doing this and here we test two methods for selecting clones. Both of these methods meet this requirement for the new population.

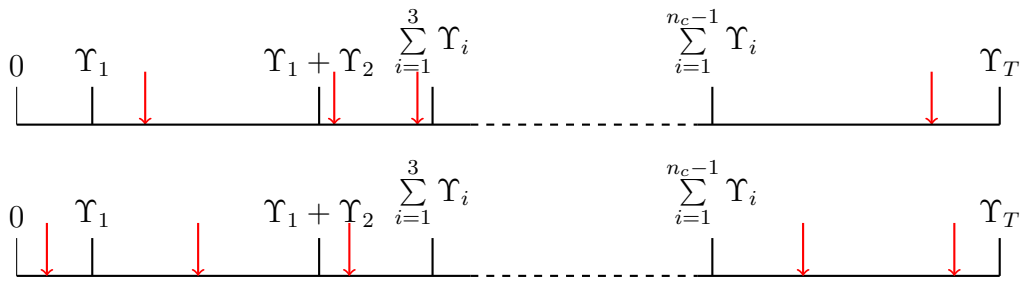


Figure 4.1: Example of the number line associated with the cloning factors $\Upsilon^\beta(\Theta_i)$ used in the clone selection methods. The red arrows indicate some of the points α_j that determine which clone is selected to be cloned. The α_j values may be chosen via the [iid] method (top) or the [eq] method (bottom).

We define a binary search key which involves a set of numbers $\Xi_\beta[k]$ for $k = 0, 1, \ldots n_c$ where $\Xi_\beta[0] = 0$ and

$$\Xi_\beta[k] = \sum_{i=1}^{k} \Upsilon_{i\beta}, \tag{4.1}$$

which defines the edges of intervals within the key for $k = 1 \ldots n_c$. We then place markers on the key and the number of markers in interval i of the key determines how many times that system i is cloned as in figure 4.1. There are two methods that we use to place the markers.

The first cloning method uses markers that are identically and independently distributed uniformly on $[0, \Upsilon_T^\beta)$. We call this method the [iid] method and each marker has a position α_j on the binary search key where $j = 1 \ldots n_c$. The state of system j in the new population of systems is then a clone of system k in the old

population of systems where k satisfies

$$\sum_{i=1}^{k} \Upsilon^{\beta}(\Theta_i) \leq \alpha_j < \sum_{i=1}^{k+1} \Upsilon^{\beta}(\Theta_i). \tag{4.2}$$

The second clone selection method places markers that are equally spaced on $[0, \Upsilon_T^{\beta})$ and we call this method the [eq] method. We choose a random number d on the interval $[0, 1)$ and the marker positions are $\alpha_j = (j + d - 1)\Upsilon_T^{\beta}/n_c$. The state of system j in the new population of systems is a clone of the system k which satisfies equation (4.2). By comparison, Lecomte and Tailleur [86] use a clone selection method where the number of times that system i is cloned is $\lfloor \Upsilon^{\beta}(\Theta_i) + \varepsilon \rfloor$ where ε is uniformly distributed on $[0, 1]$. To maintain a constant population other systems are randomly chosen to be cloned or erased.

The system on which the SSEP is implemented is defined by four components. One of these is an array of particle positions where the position of a particle is the index of the site that it occupies. Another is an array of particle freedoms where the freedom of a particle is the number of empty sites neighbouring the site that it occupies. The other two components are an array of site occupancies, the index of the particle that occupies each site and the escape rate of the system. When a site is empty its site occupancy takes a value of -1.

A system is cloned by copying each of its four components: particle positions, site occupancies, particle freedoms and the escape rate to the new set of systems. An alternative method to copying these four components to clone a system would be to copy just one of them such as the particle positions and use this to reconstruct the other components. Simulations showed, however, that for the computational approaches that we considered, this method was slower than copying all four components. After t_{obs} units of time (M cloning intervals) we compute the large deviation function $\psi(s) = \ln[Z(s, t_{\text{obs}})]/t_{\text{obs}}$.

4.2 Effect of Cloning Method and Choice of Δt

We have defined two clone selection methods in subsection 4.1. We seek to determine which of these methods is preferable in terms of accuracy and speed. To do this we define a numerical estimator for $\mathcal{K}$ which is

$$\hat{k}_L(\lambda) = \frac{1}{L n_c t_{\text{obs}}} \sum_{i=1}^{n_c} \sum_{\beta=1}^{M} K^{\beta}(\hat{\Theta}_i), \tag{4.3}$$

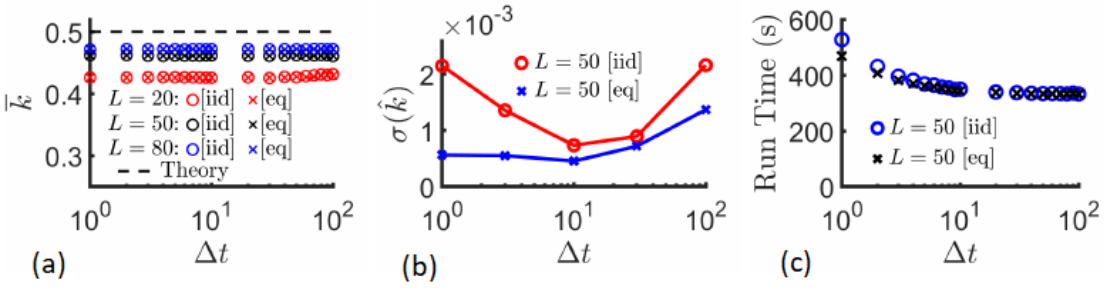


Figure 4.2: Dependence of the algorithm's results on the size of the cloning interval Δt and the clone selection mechanism ([eq] or [iid]). Results are obtained for the SSEP at a bias $\lambda = 28$ and each of the two clone selection methods [iid] (circles) and [eq] (crosses). In each case $t_{\text{obs}} = 10^4$ and $n_{\text{c}} = 10^5$, the parameters required for convergence. (a) The average of the estimator $\bar{k}$ (as defined in definition (4.4)) as the size of the cloning interval Δt is varied. Results obtained for systems of size $L = 20, 50, 80$. (b) Standard deviations in the estimator $\sigma(\hat{k})$ (as defined in definition (4.5)) measured over 100 independent realizations as the size of the cloning interval Δt is varied. Results obtained for one system size $L = 50$. (c) The run time of the code as the size of the cloning interval Δt is varied. These results are obtained for the OpenMP code as defined in section 6.2 for one system size $L = 50$.

which we obtain by copying the accumulated value $\sum_\beta K^\beta(\hat{\Theta}_i)$ of the observable during the cloning stage of the algorithm. As can be seen in figure 3.4 the directly measured activity is a more revealing measure of errors than the large deviation function.

To obtain an accurate measure of the activity we average the estimator across several realizations. We define

$$\bar{k}_L(\lambda) = \frac{1}{R}\sum_{r=1}^{R} \hat{k}_L^r(\lambda), \tag{4.4}$$

as the average of the estimator where $\hat{k}_L^r$ is the estimator associated with realization r of R independent realizations indexed $r = 1, 2, \ldots R$. When we are considering the average of the activity estimator in figure 4.2a we see that the [eq] and [iid] cloning methods give systematically the same value. We find that as the size of the cloning interval is varied, the value of the activity stays the same. These observations are true for all three values of system size $L = 20, 50, 80$ with the exception of a slight dependence of the systematic values of activity on Δt when $L = 20$ at large values of Δt. This may be due to the systematic errors becoming bigger as the cloning interval becomes larger and the frequency of cloning decreases. As the system size is increased we see the observed activity move closer to the value predicted by the

large-L theory.

We also measure the statistical errors in our results by defining the variance of the estimator

$$\Delta k_L(\lambda)^2 = \frac{1}{R}\sum_{r=1}^{R}\left[\hat{k}_L^r(\lambda) - \bar{k}_L(\lambda)\right]^2 \tag{4.5}$$

across the independent simulations. Consequently the standard deviation is defined as $\sigma(\hat{k}) = [\Delta k_L(\lambda)^2]^{1/2}$. When we investigate the size of statistical errors in activity we vary the size of the cloning interval. We see that for the [iid] method they decrease to a minimum when $\Delta t = 10$ in figure 4.2b and for the [eq] method they stay approximately constant up to $\Delta t = 10$. As we increase the size of the cloning interval beyond this the errors increase for both clone selection methods. The effect of the size of the cloning interval on the accuracy of the algorithm is not yet fully understood [95]. We find for all values of Δt that the [eq] method has smaller statistical errors than the [iid] method. This is because under the [eq] method the number of times that system i is cloned is proportional to its associated $\Upsilon_{i\beta}$ value with a maximum variation of ± 1. Under the [iid] method, however, there is more variation in the number of times that a system can be cloned.

As well as being more accurate than the [iid] method we observe in figure 4.2c that the [eq] method is marginally quicker. At values of the cloning interval $\Delta t \leq 10$ where the [eq] method is measurably quicker, the run time of the code decreases with Δt when using either clone selection procedure. We find that the run times decrease up to $\Delta t = 10$ at which point they reach a constant value that stays the same as the size of the cloning interval Δt is increased further. Our conclusion is that the [eq] method is the preferable clone selection method and that $\Delta t = 10$ is the optimal size of cloning interval as above this value of Δt we have larger statistical errors and no change in run time but below this value of Δt we have larger run times and no change in the size of the statistical errors. The accuracy and run times are both optimal for both clone selection methods at $\Delta t = 10$. There may not be a size of cloning interval for which this is the case for all processes.

4.3 Convergence

4.3.1 Time t_{obs} Convergence

We first measure how many units of time are required for the algorithm to converge to accurate results. In their two-part paper [95, 64], Nemoto, Hidalgo and Lecomte investigate the rate at which an estimator of the large deviation function converges

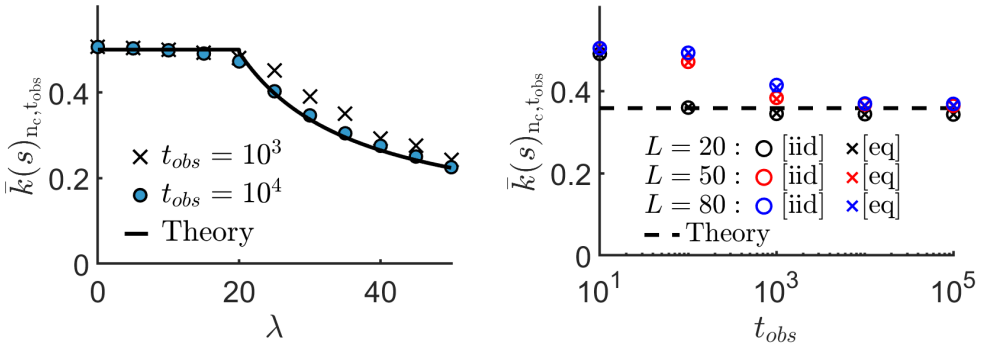


Figure 4.3: Time convergence with respect to t_{obs} of the average of the estimator $\bar{k}$ as defined in definition (4.4). When $L = 20, 50$, the number of clones $n_{\text{c}} = 10^5$ and when $L = 80$, the number of clones $n_{\text{c}} = 10^6$. The cloning interval is of size $\Delta t = 10$. (a) Results obtained at various values of the bias λ and at one system size $L = 50$. (b) The results are obtained at one value of the bias $\lambda = 28$ for systems of size $L = 20, 50, 80$ and are averaged over 10 independent simulations.

towards its true value. They have obtained the finite-time scaling behaviour of an estimator of the large deviation function so that they may attain more reliable results from the algorithm. They find that in order for the estimator to represent the actual value, corrections of order $1/t_{\text{obs}}$ in the value of the estimator have to be considered. They also find that as a power law, the scalings which dictate convergence to the infinite-time limit present no characteristic time above which the corrections would be negligible. We measure how many units of time are required for the algorithm to converge by fitting a curve

$$\bar{k}(\lambda)_{n_{\text{c}},t_{\text{obs}}} = k_\infty + A/t_{\text{obs}} \tag{4.6}$$

to our data where k_∞ is the value of $\bar{k}(\lambda)$ as $t_{\text{obs}} \to \infty$ and A is a measure of the rate of convergence. We expect A to depend on n_{c}. Hidalgo [62] has proceeded to determine that in the large-L limit, the $1/t_{\text{obs}}$ scaling ceases to be valid. The algorithm has power law corrections in t_{obs} because errors induced by the transient regimes have contributions to the algorithmic value of order $1/t_{\text{obs}}$ and this leads to errors of order $1/t_{\text{obs}}$. These are errors in the algorithmic value itself. When we obtain algorithmic results, if we have a large enough population size n_{c} then we make an accurate computation of $\ln Z(s, t_{\text{obs}})$ which we use to make an approximation to $\psi(s)$. When t_{obs} is large this approximation is accurate.

The value of the activity k_∞ as $t_{\text{obs}} \to \infty$ that we obtain by fitting a curve through the data in figure 4.3 allows us to make an estimate of the systematic error

$\log \lvert t_{\text{obs}} \rvert$	$L = 20$	$L = 50$	$L = 80$
1	3.0×10^{-1}	2.8×10^{-1}	2.8×10^{-1}
2	4.8×10^{-2}	2.3×10^{-1}	2.7×10^{-1}
3	7.9×10^{-3}	4.7×10^{-2}	1.2×10^{-1}
4	1.2×10^{-2}	2.9×10^{-3}	1.2×10^{-2}
5	3.1×10^{-4}	2.3×10^{-3}	2.3×10^{-3}

Table 4.1: Relative error in activity at $sL^2 = 28$ when varying the number of units of time. These results correspond to the values in figure 4.3b and were obtained with $n_c = 10^5$ ($L = 20, 50$), $n_c = 10^6$ ($L = 80$) clones and the [eq] clone selection method.

in the algorithmic values by calculating their difference to the infinite t_{obs} value. The magnitudes of these errors are shown in table 4.1 for systems of size $L = 20, 50, 80$. As in our paper [19] we assert a criterion of 2% as the maximum error allowed between the results from the algorithm and the infinite-time value k_∞ obtained from the curve fit. We choose 2% as our criterion because we find that when we are in this regime, the errors are small enough that they follow $1/t$ and $1/n_c$ asymptotic expansions. If we were to choose a criterion of 20%, we may find that errors that follow asymptotic expansions in $1/t^2$ or $1/n_c^2$ would not be negligible. Choosing a smaller criterion such as 0.2% would be numerically expensive and would not provide much more information. The 2% criterion provides us with a value for the number of units of time t_{obs} at which to obtain results from the algorithm. Within the 2% error tolerance we use as few units of time as we can to keep the run time of the code as low as possible.

In figure 4.3 we have also plotted the estimator $\bar{k}$ against the bias λ for two values of the t_{obs} when $L = 50$. At values of the bias up to and including the phase transition at $\lambda = 2\pi^2$, $t_{\text{obs}} = 10^3$ appears to be a sufficient number of units of time for the algorithm to converge for this system size. At larger biases, however, we clearly see that at least 10^4 units of time are needed for convergence as the values of the estimator have a smaller value than for 10^3 units of time and are closer to the $L \to \infty$ theory value. The biggest difference between the results for $t_{\text{obs}} = 10^3$ and $t_{\text{obs}} = 10^4$ is around the maximum susceptibility ($sL^2 = \lambda_{\max} \approx 28$). This suggests that the algorithm is hardest to converge around the maximum susceptibility because trajectories at the maximum susceptibility have the longest timescales associated with them. This is why we have obtained our other results in figure 4.3 and in table 4.1 at $sL^2 = 28$.

In table 4.1, the results that we have obtained are around the peak in susceptibility and the data shows us that $t_{\text{obs}} = 10^3$ units of time is the smallest number of units of time that we can use when $L = 20$ to meet our error tolerance of 2%.

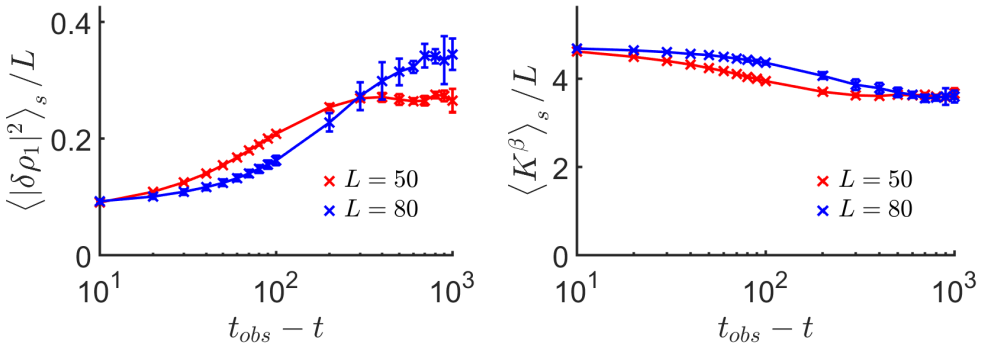


Figure 4.4: The p_{ave} measure of variables around the final transient regime for system sizes $L = 50$ (red) and $L = 80$ (blue) averaged over 5 repeats. The results are scaled by the system size L and are obtained at various values of $t_{\text{obs}} - t$. The algorithm is run for $t_{\text{obs}} = 10^4$ units of time on $n_{\text{c}} = 10^6$ systems with cloning intervals of size $\Delta t = 10$ at a bias $sL^2 = 28$. (a) The square of the first Fourier component of the density $|\delta\rho_1|^2$. (b) The activity per cloning interval K^β.

For systems of size $L = 50, 80$ we find that $t_{\text{obs}} = 10^4$ units of time is the amount required to meet our error tolerance. The number of units of time required for convergence increases with L because larger systems have longer time scales associated with them. This is because the slowest time scale in the system is associated with wavelengths of order L and is expected to scale quickly with system size. These results and those in figure 4.3 are obtained at the values of n_{c} required for the algorithm to have converged (see subsection 4.3.2).

The definition of observable averages that we use in definition (2.5) is known [70] to be dependent on the time t at which it is measured when $s \neq 0$ as discussed around definitions (2.26) in section 2.5. The trajectories that the algorithm samples have transient regimes at initial and final times whose timescale is of length τ. Between these transient regimes is a time-translation invariant (TTI) regime in which the observable distributions do not depend on t.

The results in figure 4.4 are for two observables: the first Fourier component of the density $\langle |\delta\rho_1|^2 \rangle_s / L$ measured at one instant in time and the activity per cloning interval $\langle K^\beta \rangle_s / L$ measuring across the preceding cloning interval. They show two important features associated with the transient regime that exists immediately before the final time t_{obs}. The first is the transient time scale τ which is the amount of time between the system existing in the TTI regime and the final time t_{obs}. The second is the difference between the average value of the observable in the TTI regime F_∞ and the average value of the observable at the final time.

For both observables the size of the time scale is the same $\tau \simeq 300$ when

$L = 50$ and $\tau \simeq 800$ when $L = 80$. To obtain an accurate estimate of $\mathcal{K}$ we require that $t_{\text{obs}} \gg \tau$. In this regime the sum in equation (4.3) is dominated by contributions from terms in the TTI regime. Alternatively one could estimate $\mathcal{K}$ from the plateau value of $\left\langle K^{\beta} \right\rangle_s / L$ by ignoring the transient terms in the definition (4.3) of $\hat{k}$. This may be a useful strategy for future applications of the algorithm.

Also of interest is the change in the value of the observable as t is varied from the TTI regime where the value of the the observable is F_∞ as defined in definitions (2.26) to t_{obs}. We observe in figure 4.4 that for both system sizes the value of the first Fourier component of density $|\delta\rho_1|^2_\infty / L$ that we measure in the TTI regime changes by more than a factor of 2 to the value that we measure at the final time. In contrast there are much smaller fractional changes in the amount of activity per cloning interval K^β_∞ in the TTI regime to the amount of activity per cloning interval at the final time.

This indicates that the first Fourier component of density responds much more strongly to the biasing field s, λ than the activity itself. The first Fourier component of density is related to the clustering of particles and the effect of the bias on this variable can be seen in figures 3.2 and 3.3 where the particles rapidly cluster, particularly above the phase transition. To understand further the evolution of observables over time we also study the autocorrelation function. The autocorrelation of the most relevant observable limits the convergence time of the cloning algorithm. We define the autocorrelation function of the observable F as

$$C_F(t, t') = \frac{\langle F_{t'} F_t \rangle_s - \langle F_{t'} \rangle_s \langle F_t \rangle_s}{c_F(t')}, \tag{4.7}$$

where $c_F(t') = \langle F_{t'} F_t \rangle_s - \langle F_{t'} \rangle_s \langle F_t \rangle_s$ is a normalisation factor that ensures that $C_F(t, t) = 1$ and t_1, t_2 might both be far from the boundary or one of them might be the final time. As the transient timescale τ is comparable with the inverse of the spectral gap of the stochastic process [21] one expects $C_F(t, t')$ to be small if $t - t' \gg \tau$. The time it takes for this function to decay to zero informs us of how important that the early time values of an observable are to later times and hence how many units of time that we need to run the algorithm for to assure that the effect of the early times on the results that we obtain are small.

The autocorrelation of the variables in figure 4.5 are always for $t' = t_{\text{obs}}$. This is the value of t' for which good statistics are most easily obtained. For the first Fourier component of density $|\delta\rho_1|^2$ the autocorrelation $C_F(t, t_{\text{obs}})$ with respect to the final time t_{obs} is close to a value of 1 until close to the TTI regime $t_{\text{obs}} - t \approx \tau$. After this the density fluctuations at the two times decorrelate and the

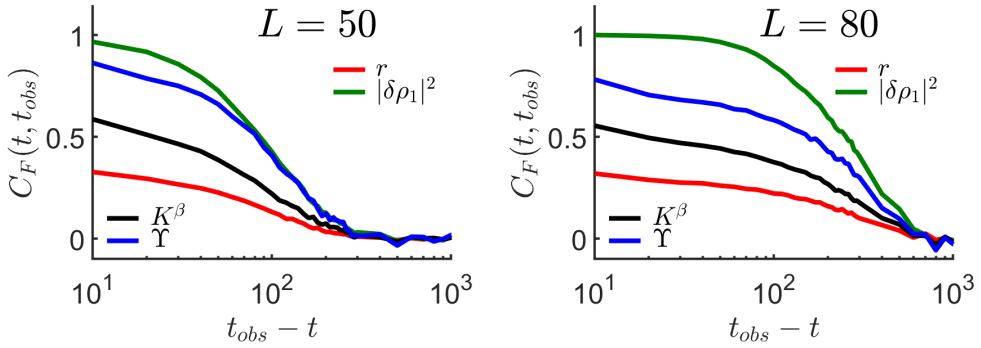


Figure 4.5: Autocorrelation Function as defined in definition (4.7) of variables with respect to the final time t_{obs} for systems of size $L = 50, 80$. The algorithm is run for $t_{\text{obs}} = 10^4$ units of time on $n_{\text{c}} = 10^6$ systems with cloning intervals of size $\Delta t = 10$ at a bias $sL^2 = 28$. The variables are the escape rate r, the square of the first Fourier component of density $|\delta\rho_1|^2$, the activity per cloning interval K^β and the cloning weight Υ. The normalisation coefficients $c_{\mathcal{O}}(t')$ are presented in table 4.2.

$\mathcal{O}$	$L = 50$	$L = 80$
r	1.49×10^{1}	2.35×10^{1}
$\|\delta\rho_1\|^2$	7.01×10^{2}	2.44×10^{3}
K^β	9.46×10^{2}	1.44×10^{3}
Υ	6.93×10^{-4}	8.33×10^{-4}

Table 4.2: Normalisation coefficients $c_{\mathcal{O}}(t')$ of various observables at $sL^2 = 28$ for systems of size $L = 50, 80$. The algorithm is run for $t_{\text{obs}} = 10^4$ units of time on $n_{\text{c}} = 10^6$ systems with cloning intervals of $\Delta t = 10$ units of time.

autocorrelation decays to 0.

For observables such as the activity per cloning interval K^β the autocorrelation $C_F(t, t_{\text{obs}})$ with respect to the final time t_{obs} is significantly less than 1 already at $t_{\text{obs}} - t = 10$. This indicates that the activity fluctuations also have a fast timescale associated with them with autocorrelations that decay quickly as well as a slow timescale that seems to be correlated with a slow decay of large density fluctuations. The decrease in the autocorrelation function over this very quick time scale varies greatly between observables. To order of magnitude, 10^3 units of time is sufficient for this autocorrelation to decay to 0 for systems with 50 sites and with 80 sites and this time scale is very similar for all of the observables.

The autocorrelations in activity play a role in the theory. The susceptibility

$$\chi \sim L^{-1} c_{K^\beta}(t_{\text{obs}}) \int_0^{t_{\text{obs}}} C_{K^\beta}(t, t_{\text{obs}}) \mathrm{d}t, \tag{4.8}$$

[48] so χ is large if the prefactor c_{K^β} is large or if the function C_{K^β} decays to zero slowly. When the latter happens the integral decays slowly to zero and it implies that the transient time scale is long. The data in table 4.2 suggests that the prefactor c_{K^β} is approximately proportional to L for the two system sizes that we have considered. The transient timescale τ is expected to scale as L^2 as diffusive decay associated with wavelengths of order L is the slowest time scale in the system. This suggests that the value of χ should increase quickly with L and this is in agreement with the heights of the peaks in χ that we observe in figure 3.4.

The key points of our analysis of the time convergence are that the longest relaxation time in the system determines the convergence with respect to t_{obs} and that these relaxation times can be explicitly determined by computing the evolution over time of the average value of observables and their autocorrelations. Making these measurements as we have done in figures 4.4 and 4.5 is also useful for revealing important physical effects in the biased trajectories that we are interested in. Above the phase transition $\lambda \gg 2\pi^2$, the main physical effect is that density ceases to be inhomogeneous on a macroscopic scale and so $\langle|\delta\rho_1|^2\rangle_s$ diverges with the system size L. There is a slow time scale associated with this inhomogeneity which leads to a large susceptibility χ and scales as $\tau \sim L^2$. The long time scale requires a large observation time t_{obs} for the cloning algorithm to be run for. This is because we require that $t_{\text{obs}} \gg \tau$ to obtain accurate values from the algorithm so that the values in the transient regimes do not have a large effect on the values obtained from the algorithm.

4.3.2 Population Size n_c Convergence

To obtain an accurate estimation of $\langle\exp(-sA_t)\rangle$ (see section 2.1) we need to simulate enough systems to sufficiently sample the trajectory space and hence gain access to the trajectories that produce the rare events of interest and have the largest contributions to $\langle\exp(-sA_t)\rangle$. One observation made by Nemoto, Hidalgo and Lecomte [64] is the finite-population convergence of an estimator of the large deviation function towards its true value. For the estimator to represent the actual value, corrections of order $1/n_{\text{c}}$ in the value of the estimator have to be considered. We fit a curve [95, 64]

$$\bar{k}(\lambda)_{n_{\text{c}},t_{\text{obs}}} = k_\infty + A/n_{\text{c}} \tag{4.9}$$

to our data where k_∞ is the value of $\bar{k}(\lambda)$ as $n_{\text{c}} \to \infty$ and A is a measure of the rate of convergence. These are different quantities to k_∞ and A in equation 4.6 and here we expect A to depend on t_{obs}. We ignore the higher order terms $\mathcal{O}(n_{\text{c}}^{-2})$ in the

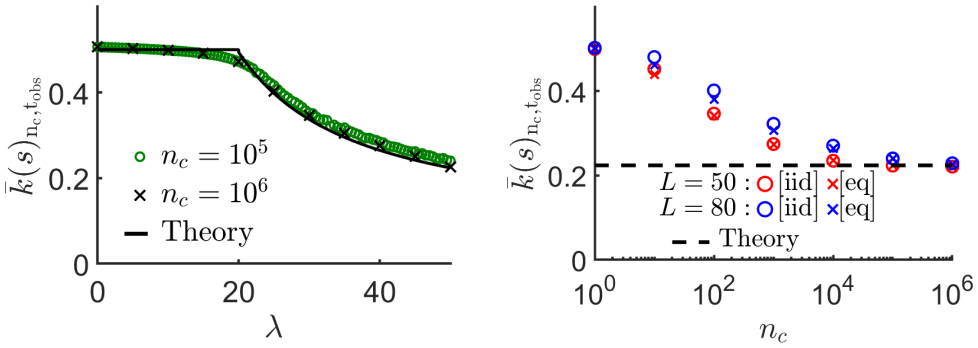


Figure 4.6: Clone convergence with respect to n_c of the average of the estimator $\bar{k}$ as defined in definition (4.4). The number of units of time $t_{obs} = 10^4$ and the cloning interval is of size $\Delta t = 10$. (a) Results obtained at various values of the bias λ and at one system size $L = 50$. (b) The results are obtained at one value of the bias $\lambda = 50$ for system sizes of size $L = 50, 80$ and are averaged over 10 independent simulations.

$\log \lvert n_c \rvert$	$L = 50$	$L = 80$
0	5.6×10^{-1}	5.4×10^{-1}
1	5.0×10^{-1}	5.0×10^{-1}
2	3.5×10^{-1}	3.9×10^{-1}
3	1.9×10^{-1}	2.5×10^{-1}
4	6.5×10^{-2}	1.2×10^{-1}
5	1.3×10^{-2}	3.5×10^{-2}
6	5.2×10^{-3}	1.8×10^{-2}

Table 4.3: Relative error in activity at $sL^2 = 50$ when varying the number of clones. These results correspond to the values in figure 4.6b and were obtained with $t_{obs} = 10^4$ units of time and the [eq] clone selection method.

asymptotic prediction [95]. Hidalgo has proceeded to determine that in the large-L limit, the $1/n_c$ scaling ceases to be valid. We can now make an estimate of how many clones are required to obtain a small error between the algorithmic value and the true value. We have chosen as our criterion a maximum 2% tolerance in the error between these two values to determine the number of clones required.

It is particularly clear in figure 4.6 that above the phase transition ($\lambda > 2\pi^2$) more systems are needed than below the phase transition. When looking at values of λ above the phase transition we observe that with $n_c = 10^6$ systems the activity estimator is less than and closer to the theory line than with $n_c = 10^5$ systems. At values of λ below the phase transition, the estimator values are the same for $n_c = 10^5$ clones and $n_c = 10^6$ clones. This is because as the bias λ increases the rarity of the rare events increases and so more systems are required to access the

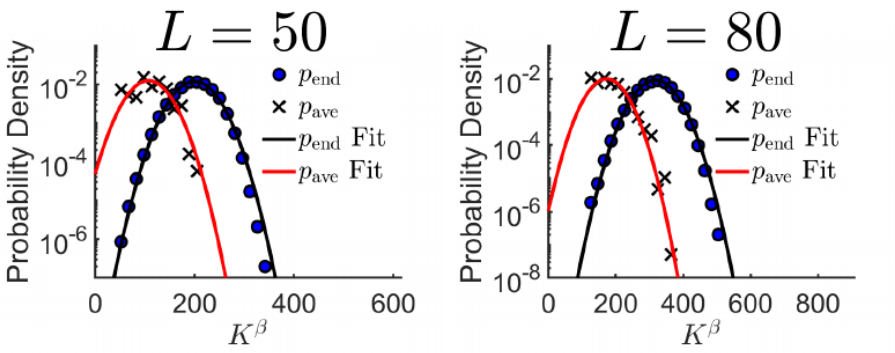


Figure 4.7: p_{ave} (crosses) and p_{end} (circles) distributions of activity per cloning interval K^β. The algorithm is run for 10^4 units of time on 10^6 systems with cloning intervals of 10 units of time. The distributions are measured at $t = 9700$ (the 970^{th} cloning interval) at a bias $sL^2 = 50$.

corresponding rarer trajectories. The figure shows that the numerical results do not align exactly with the theory line. This is due to the fact that the theory line applies as $L \to \infty$ and we are looking at a finite value of L. In figure 4.6 it is apparent that more systems are required for the algorithm's convergence as the system size increases. This is due to the fact that the bigger the system is the more possible configurations there are for the system to exist in and the less likely the rare events are. The sizes of the errors as the number of clones is varied are shown in table 4.3.

The p_{end} distribution defined in definition (2.29) in section 2.5 is sampled directly by the cloning algorithm. The p_{ave} distribution defined in definition (2.28) in section 2.5 is attained via a form of importance sampling from the p_{end} distribution. This means that data is only available for p_{ave} at values sampled by the p_{end} distribution. Since all ancestors must come from the current population we must obtain a representative sample from p_{ave} by importance sampling from p_{end}. Whether the algorithm achieves this depends on the overlap of the two distributions and the size of the population n_{c}. Unfortunately the two distributions are (much) too high-dimensional to visualise directly. It is therefore useful to examine the distributions of particular observables. One seeks observables for which p_{ave} and p_{end} have minimal overlap, since this will limit the effectiveness of the algorithm.

To obtain accurate results from the algorithm we must sample all systems that have a significant contribution to the measured value of $\langle \exp(-sA_{t_{\text{obs}}}) \rangle$ and hence dominate the final population. These are the systems that dominate the p_{ave} distribution. When the two distributions are far apart and do not overlap greatly then a large number of systems are required to sample p_{ave} sufficiently. This is one of the important bounds on how many systems are required for the algorithm

to generate accurate results. When we obtain results in figure 4.6 at values of n_c that are too small for the results to have converged, under sampling of the p_{ave} distribution is an important source of the systematic errors.

To estimate the magnitude of this effect (see also [68]) for convenience we consider the distributions of the activity per cloning interval K^β and we suppose that p_{ave} and p_{end} have approximately Gaussian distributions with the same variance σ^2. The variables μ_{ave} and μ_{end}, respectively, are defined as the mean values of the two distributions. We also define a scaled version of the complementary error function

$$G_\sigma(z) = \int_z^\infty g(y, 0, \sigma)dy,$$

where

$$g(x, \mu, \sigma) = \frac{\exp(-(x-\mu)^2/2\sigma^2)}{\sqrt{2\pi\sigma^2}}$$

and x is the observable which we are measuring the distributions of. When drawing n_c systems independently from p_{end}, we expect to sample a range of values of the observable x we are measuring, $(\mu_{end} - V) < x < (\mu_{end} + V)$ so that $G_\sigma(V) = 1/n_c$. The range of values that we sample is dependent on the number of clones n_c and hence so is the variable V. The average of x with respect to p_{ave} is

$$\bar{x} = \frac{\int_{\mu_{end}-V}^{\mu_{end}+V} x g(x, \mu_{ave}, \sigma)dx}{\int_{\mu_{end}-V}^{\mu_{end}+V} g(x, \mu_{ave}, \sigma)dx} \tag{4.10}$$

and we may replace the upper limits of the integrals with infinity when $\mu_{ave} < \mu_{end}$ and σ is not too large. We define $\Delta\mu = (\mu_{end} - \mu_{ave})$ and obtain from equation (4.10) the error

$$(\bar{x} - \mu_{ave}) \approx \sigma/\sqrt{2\pi} \cdot \frac{\exp(-(V-\Delta\mu)^2/2\sigma^2)}{G_\sigma(\Delta\mu - V)}, \tag{4.11}$$

which converges to zero as $V \to +\infty$ as it should. For large n_c we have the scaling relation $V \sim \sigma\sqrt{\log|n_c|}$ so V increases slowly with the number of systems n_c. The relevant parameter for convergence is the dimensionless quantity $X = (V - \Delta\mu)/\sigma$ which is positive when the peak of the p_{ave} distribution is within the range of data. We at least require this to assure that the p_{ave} distribution is sufficiently sampled so

$$n_c \gtrsim \exp[(\Delta\mu/\sigma)^2] \tag{4.12}$$

is a requirement for convergence. It may be possible to construct a similar general argument for an improved bound by considering that we need to sample more than

just the peak of the p_{ave} distribution to obtain a representative sample. The value $\Delta\mu$ is expected to be proportional to L and by central limit theorem so is σ^2. This means that we expect the bound on n_{c} to be proportional to $\exp(\text{constant} \cdot L)$. We expect the behaviour of A in equation 4.9 to be comparable to the bound on n_{c} at the point of convergence and hence to also be proportional to $\exp(\text{constant} \cdot L)$. Based on our Gaussian fits in figure 4.7 and the condition on the lower bounds in equation 4.12 we require 2.2×10^3 systems when $L = 50$ and 4.9×10^4 systems when $L = 80$. Our criterion for convergence informs us that we require 10^5 and 10^6 systems, respectively, for convergence. These values are significantly larger than the lower bounds that we have derived, partially because we require that we sample more of the p_{ave} distribution than just the peak.

4.4 Performance Summary

In this section we have found the [eq] clone selection method to be equivalent to the [iid] clone selection method systematically, in terms of the value of activity that it produces but superior in terms of the size of the statistical errors that it generates and its computation time in terms of the run time of the code when obtaining results for activity in the SSEP. We have also concluded that a cloning interval of size $\Delta t = 10$ is optimal in terms of the run time of the code and statistical errors but that the frequency of cloning does not seem to systematically affect the values of activity generated by the algorithm.

We have obtained values for the number of units of time t_{obs} and systems n_{c} required for the algorithm to have converged when obtaining results for systems of size $L = 20, 50, 80$ when simulating the SSEP and biasing activity. We have investigated the number of units of time required for convergence in terms of how it is related to the autocorrelations and evolution over time of a few observables. Additionally, we have derived a lower bound (4.12) on the number of clones n_{c} required for convergence by studying the distributions of activity per cloning interval.

Chapter 5

Large Deviations in Fredkin Processes

Our code that implements the algorithm is designed such that it is applicable to a wide range of systems and processes. Large deviations have been observed in a range of settings such as 2-level open quantum systems [46, 93], one-dimensional systems with attractive interactions [20], rectangular networks [118] and reset processes [60, 92]. The next process that we have investigated is the Fredkin Process [105, 93] that operates on a one-dimensional lattice. This process is similar to the SSEP but the particles operate under additional restrictions. We therefore observe the effects that these restrictions have on the value of the observable and the value of the large deviation function by comparison with the results that we have obtained from the SSEP. We test whether the restrictions that govern the particle activity and its rates affect the process observables. Specifically, we consider the peaks in susceptibility and how the values of the bias at which the peaks occur and heights of the peaks are affected by additional Fredkin restrictions. Also we observe the relations between observables. We investigate how biasing one observable affects the other observables and whether biasing one of these other observables leads to the same rare events.

5.1 Fredkin Process and Relevant Observables

The Fredkin Process has been studied in great detail by Salberger and Korepin [105] where they set out the Hamiltonian of the Fredkin Process in terms of Fredkin Gates. This is the quantum model from which we describe classical stochastic Fredkin Processes. They solve their model using Catalan combinatronics in the form of random walks which exist on the upper half of a square lattice. One quantity of interest is the size of the spectral gap, this is the difference between the first two

largest eigenvalues of the transition matrix [90]. It is of interest because it is related to timescales within the process [21]. Several results on the size of the spectral gap of the Fredkin quantum spin chain Hamiltonian have been obtained by Movassagh [93].

Fredkin Processes have been studied as gapless quantum spin chains [26]. The gap is the difference between the smallest and second smallest eigenvalues of the Hamiltonian associated with the system. The gap of the Hamiltonian is positive for finite systems but tends to zero as the system size tends to infinity and this is the gapless property that they refer to in [26]. They are of interest because they exhibit long range correlations due to their occupancy rules which we do not observe in the SSEP. One dimensional processes such as Fredkin Processes are ideal for researchers to work with because they are often more tractable numerically and analytically than realistic models of quantum critical systems which are often very challenging for researchers to study due to the presence of strong interactions. The Fredkin Gates which the Hamiltonian of the Fredkin Process can be set out in terms of were first proposed by Edward Fredkin and are one of the first examples of a reversible logic operation which conserves the number of bits and for which no energy is dissipated as a result of erasure [99].

One useful representation of these processes is their Dyck Paths [105, 93]. In a Dyck Path representation such as in figure 5.1 an 'up-diagonal' (bottom-left to top-right) represents an occupied site and a 'down-diagonal' (top-left to bottom-right) represents an empty site. These correspond to Dyck Words represented by open brackets and closed brackets. At either end of the one-dimensional lattice there is an implicit occupied site (left-hand end) and an implicit empty site (right-hand end).

One of the rules that governs the Dyck Paths of Fredkin Processes is that the area beneath it is always positive. This is because a Fredkin Process's Dyck Path, which begins directly before the implicitly occupied site to the left of the left-hand boundary of the lattice, never has a height less than zero. This means that when reading from left to right there are always more occupied sites than unoccupied sites. In a Dyck Word there must be a preceding open bracket not corresponding to a succeeding closed bracket for a closed bracket to be placed. We always consider half-filling so that the Dyck Path has the same initial height as final height and so that there are an equal number of open brackets and closed brackets in the Dyck Word.

Under the Fredkin Process we consider there is exclusivity so that particles may not move to an already occupied site. Furthermore the boundary conditions

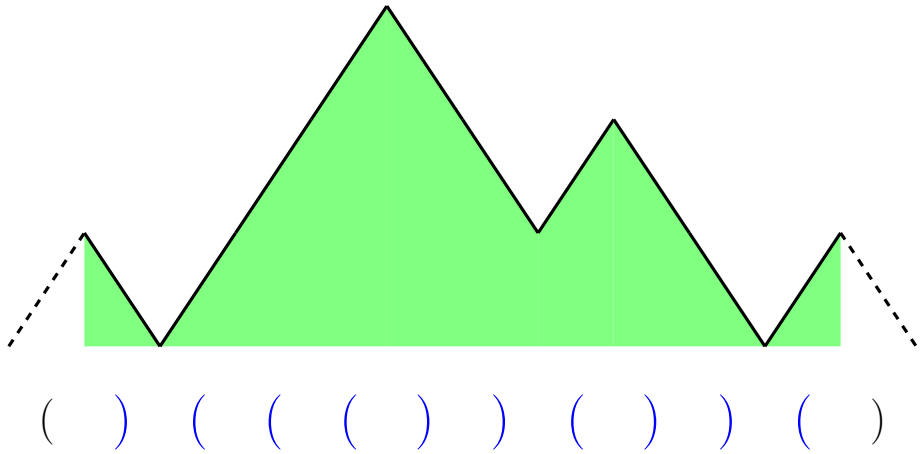


Figure 5.1: An example of a Dyck Path and its corresponding Dyck Word. The shaded green area is what we consider to be the area beneath the Dyck Path and does not include the area beneath the implicit boundary sites.

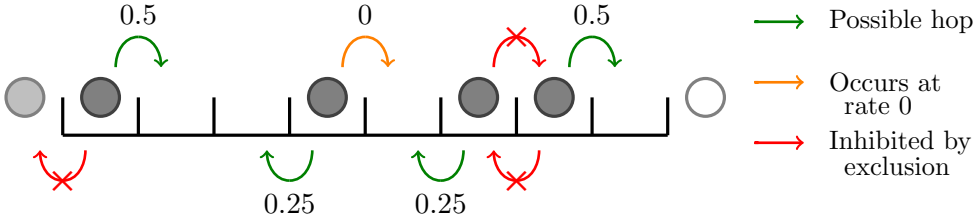


Figure 5.2: Illustration of a Fredkin Process on a one-dimensional lattice of 8 sites with non-periodic boundaries and implicit sites at either end. The left-hand implicit site is occupied and the right-hand implicit site is unoccupied.

are that the particles may not jump from either end of the lattice and there are no periodic boundary conditions. Each particle hops with rate 0.25 to neighbouring empty sites but this rate is modified by local rules. An example of this is shown in figure 5.2 and the rules are listed below.

- If the site to the left of the particle is occupied before the hop then the rate of the hop is increased by 0.25.
- If the site to the left of the particle is occupied after the hop then the rate of the hop is increased by 0.25.
- If the site to the right of the particle is occupied before the hop then the rate of the hop is decreased by 0.25.
- If the site to the right of the particle is occupied after the hop then the rate

of the hop is decreased by 0.25.

This means particles are more likely to hop to and from sites with neighbouring sites to the left of them that are occupied and less likely to hop to and from sites with neighbouring sites to the right of them that are occupied.

There are several observables that we may measure the large deviations of under the Fredkin Process. As with the SSEP we may consider activity and current. Another observable which we may consider is the area under the Dyck Path which is defined in figure 5.1 and does not include the area beneath the implicit boundary sites. By biasing the area under the Dyck Path we measure its large deviations and other observables under this bias. The occupancy rule as we read left to right makes the centre of mass under constraint and of interest.

When all of the particles are clustered to the left hand side of the lattice, the area underneath the Dyck Path is N^2 where N is the number of particles. From this we can access any combination of particles by shifting particles to the right. Each particle shift reduces the area by 2. We can state the area under the Dyck Path as $N^2 - 2h_r$ after h_r shifts to the right. To normalise such that the maximum area is 1 we have that the area is

$$\text{area} = \frac{N^2 - 2h_r}{N^2} = 1 - \frac{2h_r}{N^2}. \tag{5.1}$$

The centre of mass can be stated as $\sum_{i}^{N} p[i]/N$, the average position of the particles. In this case the position of a particle $p[i]$ is $(s[i] - 1/2)/L$ where L is the number of sites in the lattice and $s[i]$ is the site occupied by particle i when the sites are indexed $1, 2, \ldots, L$. We include a $-1/2$ so that the when the particles are clustered to the left hand side of the lattice, the centre of mass is $[N \times N/2]/NL = N/2L = 1/4$.

As particles hop to the right this increases the centre of mass by $1/(2N^2)$. Hence, the centre of mass of the particles is $1/4 + h_r/(2N^2)$. By substituting this equation for centre of mass into the equation for area beneath the Dyck Path we obtain that

$$\text{com} = \frac{1}{2} - \frac{\text{area}}{4}, \tag{5.2}$$

where com is the centre of mass and that obtaining values for fluctuations in area is equivalent to obtaining values for fluctuations in centre of mass.

One key difference between the SSEP and Fredkin Processes is that the SSEP does not necessarily have an associated Dyck Path with a positive area as the height of the Dyck Path under SSEP is free to have a value less than zero. Furthermore we know that SSEP exhibits a phase transition in activity when it is biased. We let s^* be

the value of s which maximises $\chi(s)$ and know that in the SSEP, s^* scales with L^{-2} [85]. We expect $\psi(s)$ to have an altered curvature when we impose restrictions on the allowed activity under the Fredkin Process. Hence, we expect to see alternative scalings in s^* that may not go like L^{-2}.

5.2 Large Deviations in Activity (Hops)

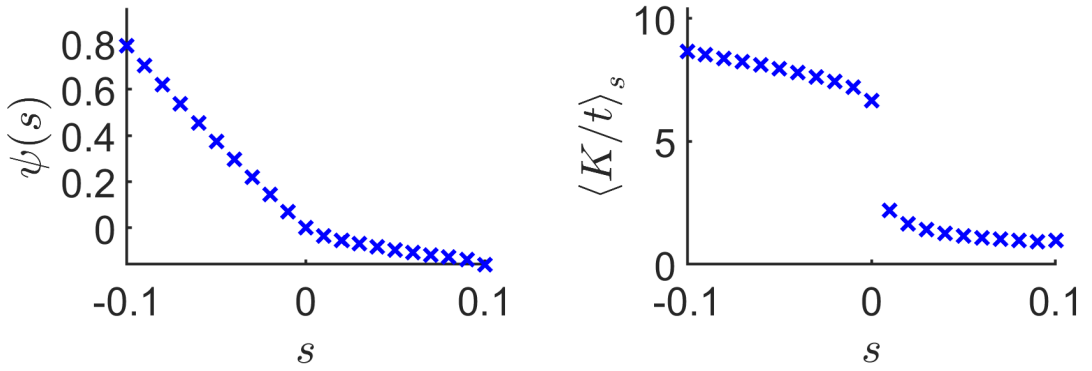
When simulating the Fredkin Process, we measure large deviations of the activity. This is a type $\mathcal{A}$ observable as the activity changes with configuration changes, however we do not modify the dynamics. This is because if we were to modify the dynamics in the same way in which we have modified the dynamics for the SSEP we would need to track the value of the escape rate. This is difficult to do in the case of the Fredkin Process because when a particle hops it does not just affect the rate at which its neighbouring particles move but the rate at which particles two sites away from it move. An additional difficulty in tracking the escape rate of Fredkin Chains is that the rule that there must be more occupied than unoccupied sites when reading across the lattice from left to right also needs to be tracked.

Therefore the cloning weights are of the form $\Phi_{i\beta} = \exp(-sA^{i\beta})$ and we calculate $\hat{\psi}_{n_c,t_{\text{obs}}}(s)$ and update $Z(s,t)$ according to the procedures for when the dynamics are not modified in sections 2.3 and 2.4. In figure 5.3 the large deviations results for Fredkin Processes show us an approximately linear region of the large deviation function at negative s and at $s = 0$ a transition to the positive s region where the large deviation function no longer follows this straight line. This corresponds to a clear step change in the value of the activity from high activity at negative s to low activity at positive s although in neither region is the value of the activity constant.

We check in figure 5.4 that our results for the Fredkin Process are giving us accurate values by comparing to some theoretic results. In figure 5.4a, we show that the algorithm data aligns with large deviations values obtained from the transition matrix [83, 36, 84, 57]. Jack and Sollich [70] define a transition matrix $\mathbb{W}(s)$ whose terms depend on the transition rates $W(C \to C')$ and escape rates $r(C)$,

$$\langle C|\mathbb{W}|C'\rangle = \begin{cases} W(C' \to C)\exp(-s\alpha(C',C)), & C \neq C', \\ -r(C), & C = C', \end{cases} \tag{5.3}$$

where $\langle A|\mathbb{W}|B\rangle$ refers to the entry of the transition matrix in row A of column B. As usual $\alpha(C,C')$ refers to the change in the value of the observable between configuration C and configuration C'. It is demonstrated by Jack and Sollich [70] that the large deviation function $\psi(s)$ can be obtained via eigendecomposition of



(a) Large Deviation Function

(b) Activity

Figure 5.3: Large deviation function and directly measured activity when biasing activity. Results obtained at $t_{\text{obs}} = 10^4$, $n_{\text{c}} = 10^5$, $L = 50$, $\Delta t = 10$ over 10 repeats with the [eq] (equal spacing) clone selection method and non-modified dynamics.

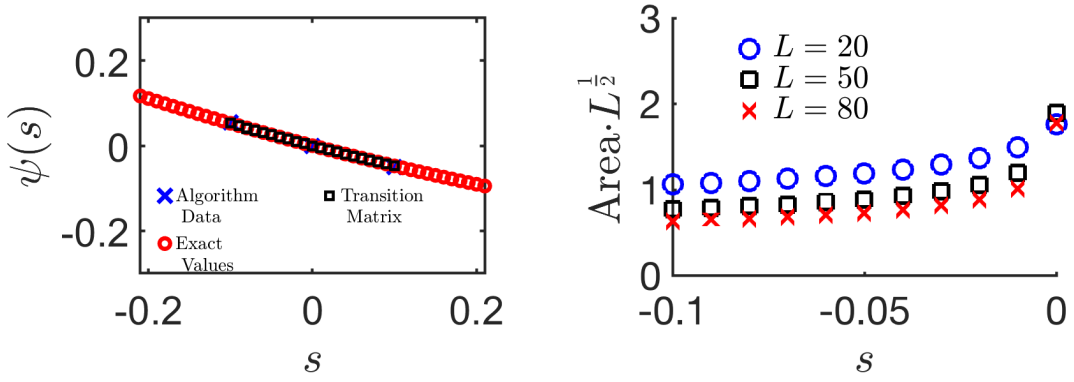
$\mathbb{W}(s)$. Specifically, by defining ω_i as the eigenvalues of $\mathbb{W}(s)$ they obtain the result

$$\psi(s) = \max_i \omega_i, \tag{5.4}$$

which we use to compute the Transition Matrix values of the large deviation function in figure 5.4. We also compare to results obtained by computing fully the biased Hamiltonian of the system which we compute by calculating its boundary and bulk terms using spin operators. The biased Hamiltonian is identical to the transition matrix $\mathbb{W}(s)$ and again we calculate its eigenvalues to obtain the large deviation function. Figure 5.4 also shows alignment between the algorithm data and these numerically exact values.

We also check that the area underneath the Dyck Path scales as expected with system size. In figure 5.4b when there is no bias, the average area scales with $L^{-1/2}$. The result for $L = 50$ at $s = 0$ is above the results for both $L = 20$ and $L = 80$ because of statistical variation. We expect from the theory of Brownian bridges [73] that the average value of the area on a unit interval $[0, 1]$ is some constant. As we scale the system length by L, the length of the Dyck Path scales by L and its height scales by making the standard Brownian rescaling $L^{1/2}$. We normalise every area that we measure by a factor of L^{-2} to ensure that the maximum area we can observe is equal to 1, so we obtain the observed $L^{-1/2}$ scaling in the average area beneath the Dyck Path. As we scale to a negative bias we see that Area$\cdot L^{1/2}$ is decreasing with L in figure 5.4b.

We then look at properties of the system under the Fredkin Process when we bias the activity. In figure 5.5 we see how the systems evolve with time at different



(a) $\psi(s)$, $L = 2$

(b) Area, $L = 50$

Figure 5.4: (a) Large deviation function comparison between data from the algorithm, exact data and results from the transition matrix for a system of size $L = 2$. (b) Directly measured area scaled by a factor of $L^{\frac{1}{2}}$ when biasing hops. Algorithmic data in both subfigures obtained at $t_{\text{obs}} = 10^4$, $n_{\text{c}} = 10^4$, $\Delta t = 10$ averaged over 10 repeats with the [eq] (equal spacing) clone selection method.

values of the bias s. These trajectories show a clustering of particles on the left boundary of the lattice which corresponds to the bottom of the trajectories in figure 5.5 when we bias towards a positive s. From figure 5.3b we know that a positive bias corresponds to a low activity. We also make a p_{ave} measure of the density profile of systems of size $L = 20$. This is done by measuring the site occupancies of each system at the end of each cloning interval.

They show that when there is no bias or a small negative bias that particles naturally cluster on the left hand side of the lattice to preserve the rule that there are more occupied than empty sites when reading from left-to-right. The panels in figure 5.6 also show that as the bias becomes more positive the particles become more clustered on the left hand boundary. A further noticable feature of the panels in figure 5.6 is their alternating structure. This is particularly clear when the particles are spread out and the sites alternate between occupied and unoccupied.

One feature of the large deviations cloning algorithm that we are using is that we may measure other observables than the one that we are biasing at a fixed value of the bias. For example, when we bias the activity we may directly measure the values of the average area beneath the Dyck Path as in figure 5.7. At negative s where we have biased to a large activity the particles are spread out and hence the area is low. As the bias s is increased to positive s the low activity corresponds to the area being maximised.

We can also measure the centre of mass varying with s. We know that when the bias is positive, the activity is low and that this corresponds to particles cluster-

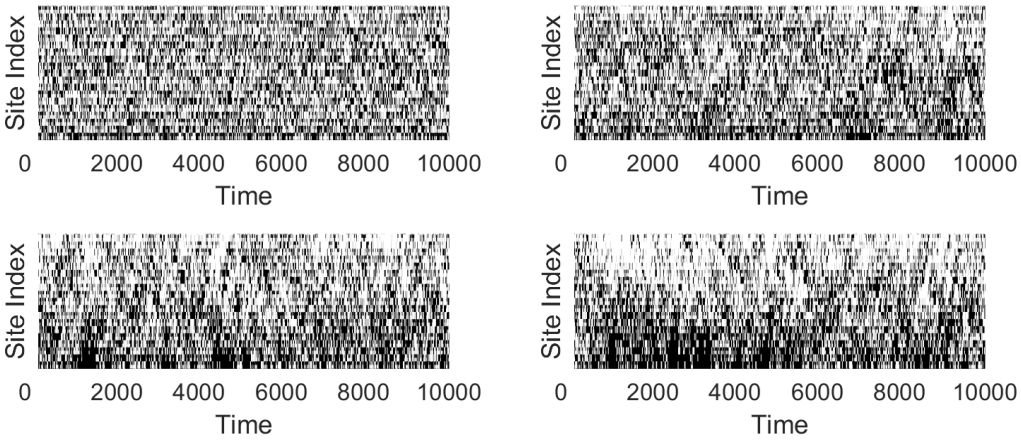


Figure 5.5: Sample trajectories of Fredkin Processes when biasing hops with $L = 20$, $t_{\text{obs}} = 10^4$, $n_{\text{c}} = 10^4$ and $\Delta t = 10$. Trajectories of systems that have index 0 at time t_{obs} when primary seed is 0. (a) $s = -0.1$; (b) $s = -0.01$; (c) $s = 0$; (d) $s = 0.01$.

ing on the left boundary. We can calculate what centre of mass that this corresponds to. There are two ways of calculating the centre of mass: either from the directly measured area (which tells us how many particle hops to the right that there are from the configuration where all particles are clustered on the left hand boundary) or from the p_{ave} measure of the density profile. We see in figure 5.8 that as expected a low activity corresponds to a low centre of mass. There is good agreement between the two methods of obtaining the centre of mass. At negative s the centre of mass approaches 0.5 which is its maximum possible value.

5.2.1 Peak in Susceptibility

The values in figure 5.3 suggest a sharp drop off in activity in the positive s regime close to $s = 0$. When we zoom in on this transition in figure 5.9 we plot our results against sL^2. We have found that when we make an sL^2 scaling of our results that different system sizes have similar large deviations as is the case in the SSEP. This can also be seen in the values of $\mathcal{K}(sL^2)$. The corresponding peak in susceptibility exists somewhere between $sL^2 = 0$ and $sL^2 = 5$ in the system sizes that we have considered $L = 20, 50, 80$.

As with the SSEP in figure 3.4, the value s^*L^2 which maximises $\mathcal{X}$ decreases in value as the system size L is increased. In the SSEP this value decreases towards the value $2\pi^2$ but under the Fredkin Process this value decreases towards a value less than $2\pi^2$ that may be zero. We also note that the peak in $\mathcal{X}$ increases in value as the system size is increased from $L = 20$ to $L = 50$ but does not increase further

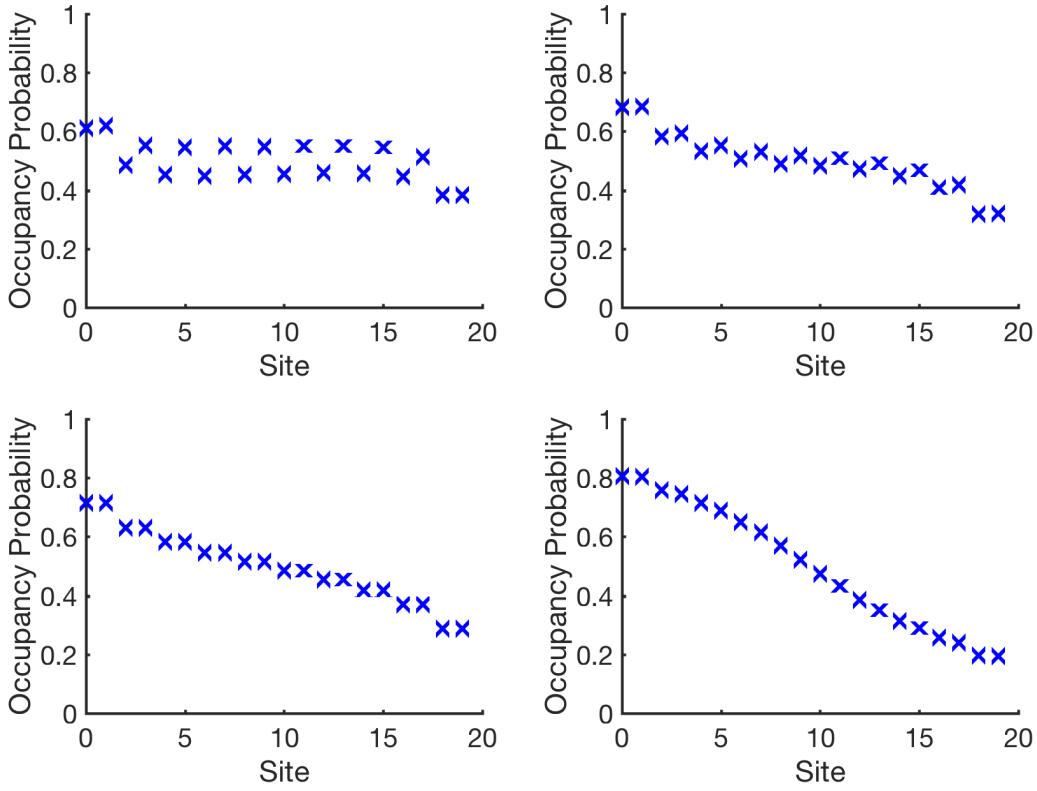
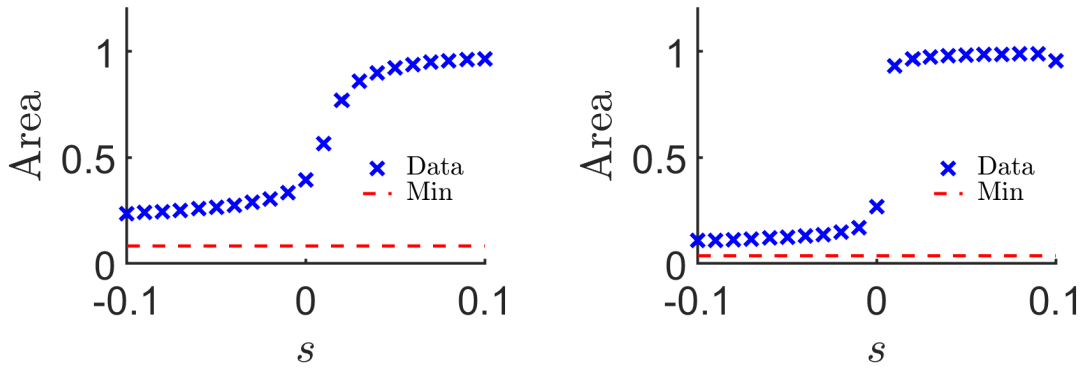


Figure 5.6: Density Profile of Fredkin Processes when biasing hops with $L = 20$, $t_{\text{obs}} = 10^4$, $n_{\text{c}} = 10^4$ and $\Delta t = 10$ over 10 repeats. (a) $s = -0.1$; (b) $s = -0.01$; (c) $s = 0$; (d) $s = 0.01$.

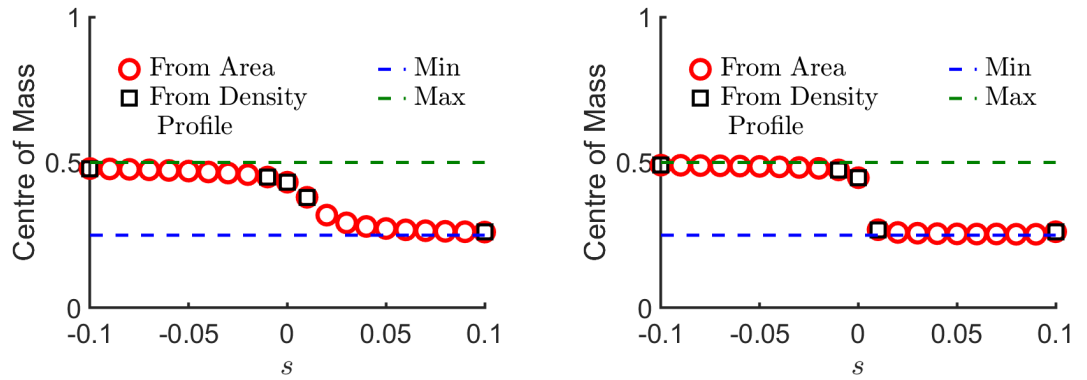
when the system size is increased further from $L = 50$ to $L = 80$ or is increasing more slowly if at all. This is in contrast to the SSEP, where the peak in $\mathcal{X}$ continues to increase in value as the system size is increased from $L = 20$ to $L = 40$ and then to $L = 80$.

When we bias activity in the SSEP as in chapter 3, the phase transition exists at a fixed value of a scaling of the bias sL^2. As discussed in section 5.1, when we impose restrictions on the activity in the SSEP as we do in the Fredkin Process, we expect to see alternative scalings of the bias at which the maximum susceptbility occurs. Figure 5.9 shows that at the sL^2 scaling of the bias, although different system sizes have similar large deviation functions and values of activity when biasing the activity in the Fredkin Process, the results do not align as clearly as they do when biasing activity in the SSEP. Furthermore, the values of sL^2 at which the peaks in susceptibility occur are not as similar for different system sizes in the Fredkin Process as they are in the SSEP. This suggests that there is a better scaling than the sL^2 scaling to plot our results against. An informative objective



(a) $L = 20$ (b) $L = 50$

Figure 5.7: Directly measured area when biasing hops. Results obtained at $t_{\text{obs}} = 10^4$, $n_{\text{c}} = 10^4$ ($L = 20$), 10^5 ($L = 50$), $\Delta t = 10$ over 10 repeats with the [eq] (equal spacing) clone selection method and non-modified dynamics. Dashed red line represents the minimum possible area for each system size.



(a) $L = 20$ (b) $L = 50$

Figure 5.8: Directly measured centre of mass when biasing hops. Results obtained at $t_{\text{obs}} = 10^4$, $n_{\text{c}} = 10^4$ ($L = 20$), 10^5 ($L = 50$), $\Delta t = 10$ over 10 repeats with the [eq] (equal spacing) clone selection method and non-modified dynamics.

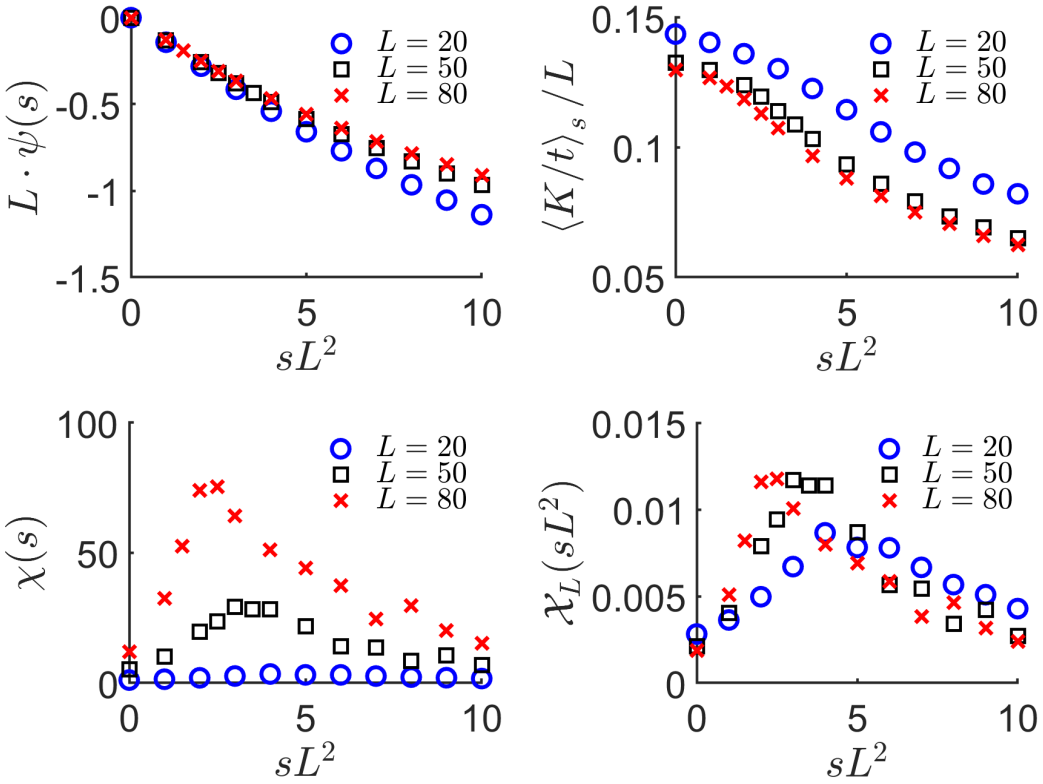


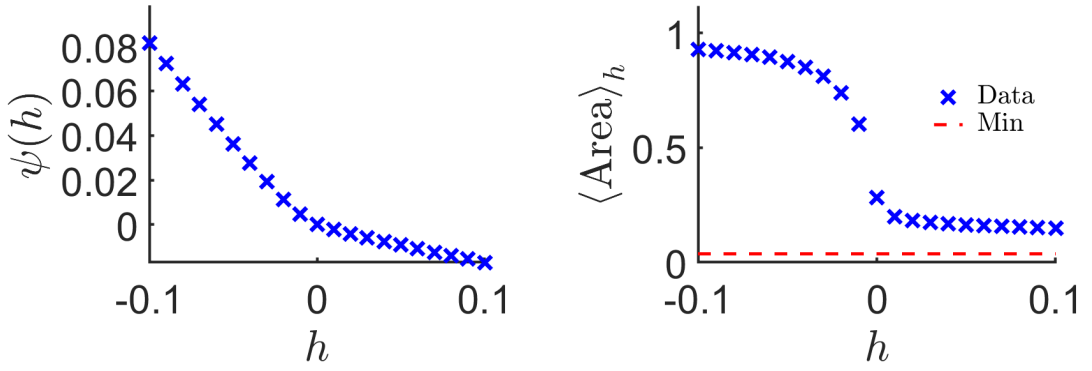
Figure 5.9: Large Deviations, activity and susceptibilty of Fredkin Processes when biasing hops with system sizes $L = 20$ ($t_{\text{obs}} = 10^4$, $n_{\text{c}} = 10^3$), $L = 50$ ($t_{\text{obs}} = 10^5$, $n_{\text{c}} = 10^4$) and $L = 80$ ($t_{\text{obs}} = 10^6$, $n_{\text{c}} = 10^4$) when $\Delta t = 10$. The results are averaged over 50, 50 and 10 repeats respectively. In figure 3.4 we show similar results for the SSEP.

of future research would be to determine the correct scaling for our results to be plotted against.

5.3 Large Deviations in Area Beneath the Dyck Path

As discussed in section 5.1 one of the observables relevant to the Fredkin Process is the area beneath the Dyck Path (see figure 5.1) sometimes simply referred to as the area. In this section we bias the area and seek to find similarities and symmetries when comparing these results with the results that we have obtained when biasing the activity. For clarity we call this bias h and obtain results at small values of the bias $-0.1 < h < 0.1$.

The Area beneath the Dyck Path is a type $\mathcal{B}$ observable as its value only



(a) Large Deviation Function

(b) Area

Figure 5.10: Large deviation function and directly measured area when biasing area. Results obtained at $t_{\text{obs}} = 10^5$, $n_{\text{c}} = 10^4$, $L = 50$, $\Delta t = 10$ over 10 repeats with the [eq] (equal spacing) clone selection method and non-modified dynamics.

depends on the state at a point in time and not its history, hence we do not modify the dynamics. The cloning weights are therefore of the form $\Phi_{i\beta} = \exp(-sA^{i\beta})$ and we calculate $\hat{\psi}_{n_{\text{c}},t_{\text{obs}}}(s)$ and update $Z(s,t)$ according to the procedures in sections 2.3 and 2.4 for when the dynamics are not modified.

Again we see a decrease in the observable value, this time area, as the bias is increased from negative to positive. As with activity, in figure 5.10 there is a kink around $h = 0$ as the value of the observable quickly drops off. This time the area seems to decrease most quickly when the bias is negative. We know that when particles are clustered to the left that this corresponds to a high area. When we look at the trajectories of the systems in figure 5.11 we see that the negative bias which produces these high area values also produces the left boundary clustering as expected.

When we look at the density profiles and centre of mass in figures 5.12 and 5.13, we also see the expected left hand clustering at a negative bias. As we bias towards low area, the system takes a centre of mass that approaches 0.5, its maximum possible value. We see that the positive bias hence produces a flatter density profile although the left to right occupancy rule cannot be broken.

5.4 Algorithm Performance

In a previous chapter (4) we have investigated how the algorithm performs at obtaining results for the SSEP. We have determined the best parameters to use in terms of speed and accuracy. We investigate in this section how the algorithm performs when we obtain results for Fredkin Processes. We consider the case where we

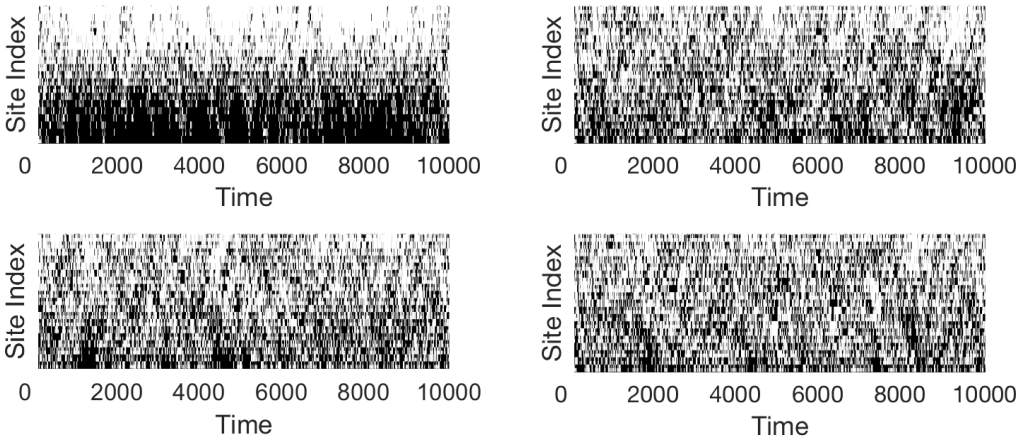


Figure 5.11: Sample trajectories of Fredkin Processes when biasing area with $L = 20$, $t_{\text{obs}} = 10^4$, $n_{\text{c}} = 10^4$ and $\Delta t = 10$. Trajectories that arrive at index 0 when primary seed is 0. (a) $h = -0.1$; (b) $h = -0.01$; (c) $h = 0$; (d) $h = 0.01$.

bias the activity but also the case where we bias the area beneath the Dyck Path. We specifically measure the number of clones and units of times required for the algorithm to converge and hence use several of the definitions used in section 4.3.

The system on which the Fredkin Chains are implemented is defined by two components. One of these is an array of particles positions where the position of a particle is the index of the site that it occupies. The other is an array of site occupancies, the index of the particle that occupies each site. When a site is empty its site occupancy takes a value of -1. A system is cloned by copying both of these components. After t_{obs} units of time (M cloning intervals) we compute the large deviation function $\psi(s) = \ln[Z(s, t_{\text{obs}})]/t_{\text{obs}}$.

5.4.1 Measuring Activity (Hops)

We first consider how many units of time t_{obs} and how many clones n_{c} are required to converge the results at various values of the bias s when $L = 50$ in figures 5.14 and 5.15, respectively at a cloning interval $\Delta t = 10$. In figure 5.14 we fit a curve of the form $k_\infty + A/t_{\text{obs}}$ as in equation (4.6) and in figure 5.15 we fit a curve of the form $k_\infty + A/n_{\text{c}}$ as in equation (4.9) at the values of the bias s that are hardest to converge. We then analyse how may units of time t_{obs} and how many clones n_{c} are required to converge the results in more detail for systems of size $L = 20, 50, 80$.

We have found in the SSEP that the phase transition is the hardest place to converge the algorithm with respect to the number of units of time. We consider systems of size $L = 20, 50, 80$ and investigate how many units of time are required

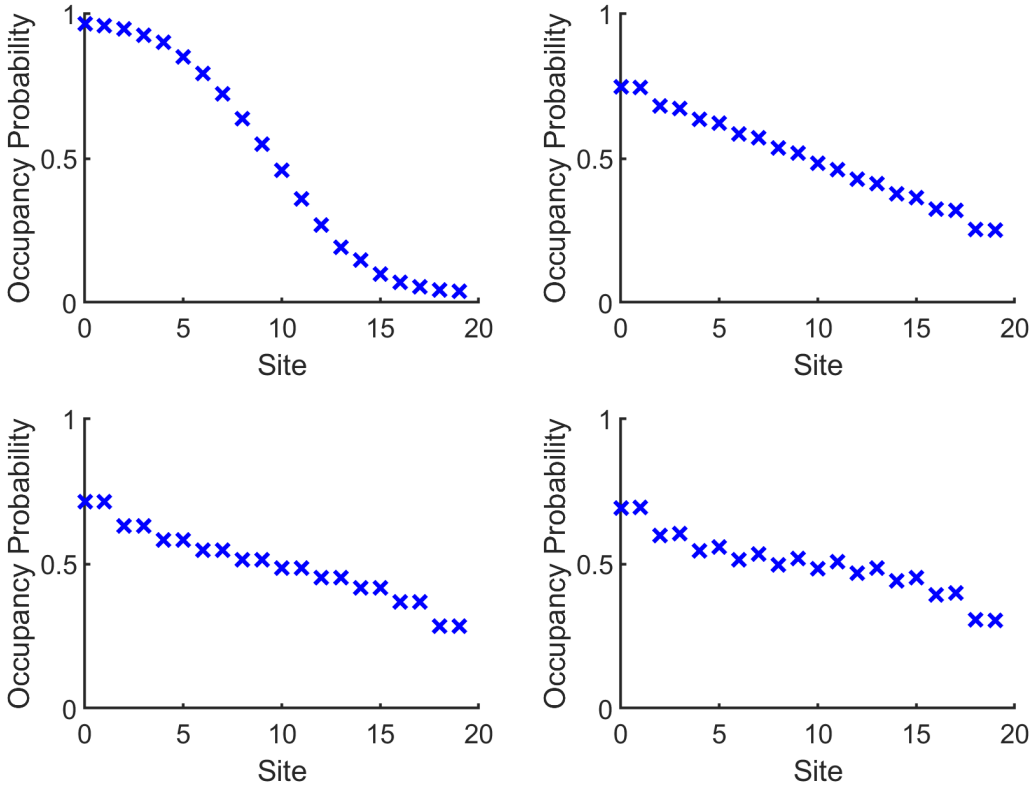
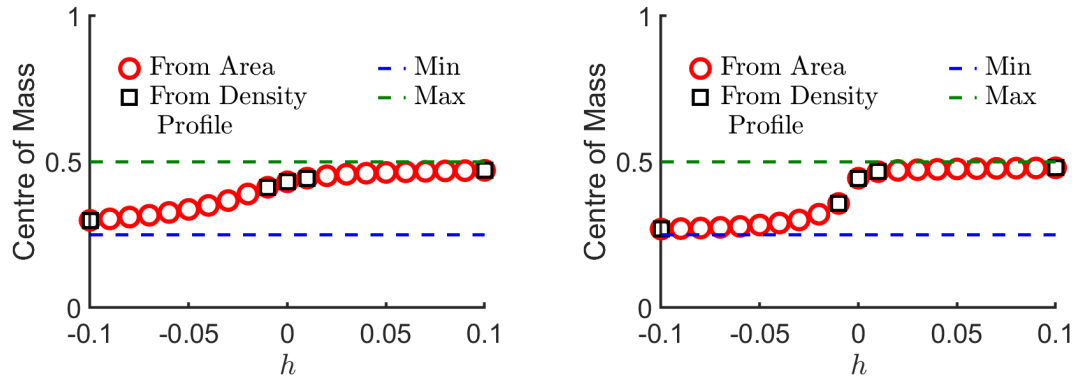


Figure 5.12: Density Profile of Fredkin Processes biasing area with $L = 20$, $t_{\text{obs}} = 10^5$ and $n_{\text{c}} = 10^3$. (a) $h = -0.1$; (b) $h = -0.01$; (c) $h = 0$; (d) $h = 0.01$. These values are obtained at $\Delta t = 10$ averaged over 10 repeats with the [eq] (equal spacing) clone selection method.

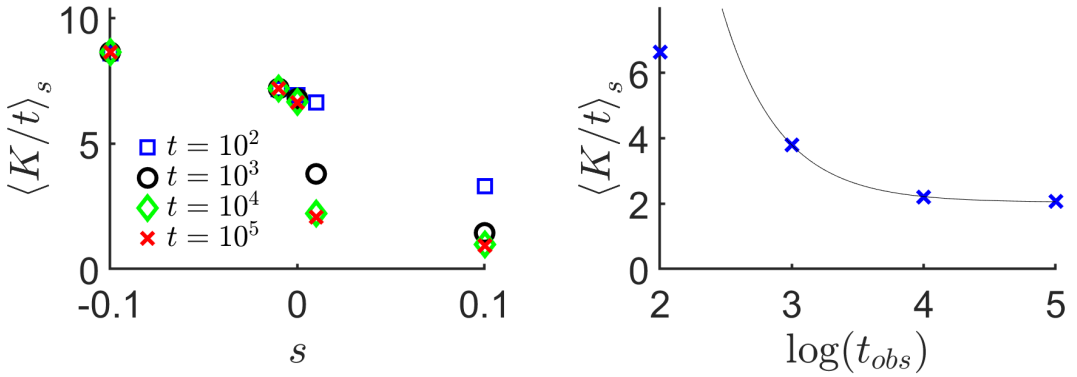
to converge the algorithm at the maximum susceptibility. In figure 5.16 we fit a curve of the form $k_\infty + A/t_{\text{obs}}$ as in equation (4.6) through data for system sizes $L = 50, 80$ as we have done for the SSEP in section 4.3. The value of k_∞ represents the value of the activity as $t_{\text{obs}} \to \infty$ and A is a fitting parameter that depends on the rate of convergence. We fit these curves through the final three data points in each case, and these are used to obtain values of k_∞ and hence the data in Table 5.1 which includes values for $L = 20$.

The curve fits provide us with an estimate of the value of activity as $t_{\text{obs}} \to \infty$. The values in table 5.1 then provide us with an estimate of the systematic error between the algorithm's value and this infinite t_{obs} value when we use a finite number of time steps. As in our paper [19] we assert a 2% criterion of the maximum error between the results from the algorithm and the results from the curve fit. This allows us to derive a value for the number of units of time to use when obtaining results around the maximum susceptibilty. As always we look to use as few units of



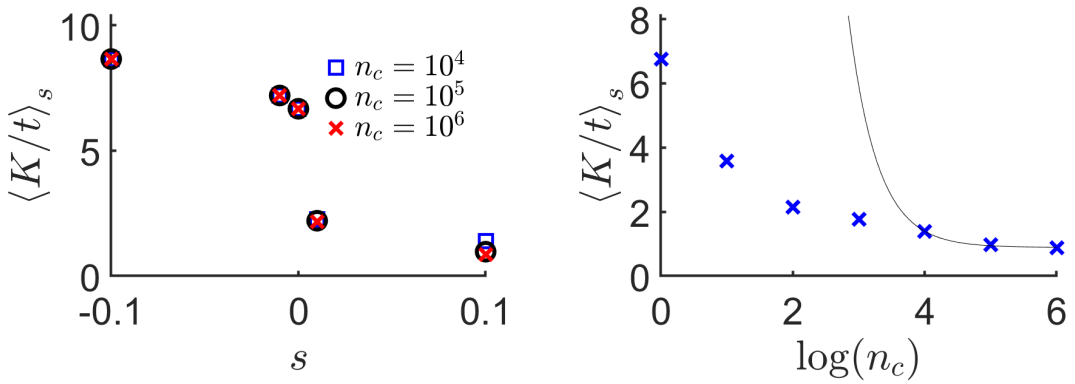
(a) $L = 20, n_c = 10^3$ (b) $L = 50, n_c = 10^4$

Figure 5.13: Centre of Mass calculated via two methods from the directly measured area and from the density profile. Results obtained at $t_{obs} = 10^5$,$\Delta t = 10$ over 10 repeats with the [eq] (equal spacing) clone selection method.



(a) Various s (b) $s = 0.01$

Figure 5.14: Time convergence of directly measured activity in Fredkin Processes when biasing the activity. Results obtained for $L = 50$ sites with $n_c = 10^5$ clones and a cloning interval $\Delta t = 10$. The results are averaged over 10 repeats and obtained with the [eq] (equal spacing) clone selection method and non-modified dynamics. (a) Results are obtained at various values of the bias s (b) The black curve is a fit through the final three data points using a least squares fitting method of the form $k_\infty + A/t_{obs}$ where $k_\infty = 2.04$ (3s.f.) and $A = 1.75 \times 10^3$ (3s.f.).



(a) Various s (b) $s = 0.1$

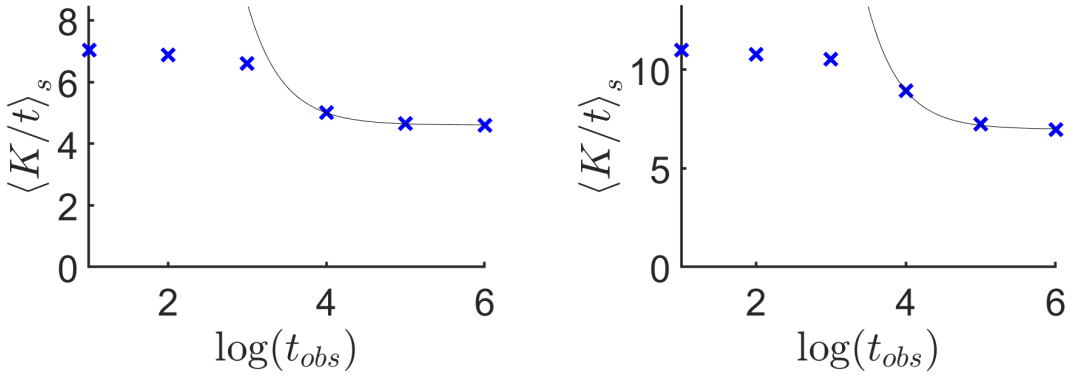
Figure 5.15: Clone convergence of directly measured activity in Fredkin Processes when biasing the activity. Results obtained for $L = 50$ sites with $t_{\text{obs}} = 10^4$ units of time and a cloning interval $\Delta t = 10$. The results are averaged over 10 repeats and obtained with the [eq] (equal spacing) clone selection method and non-modified dynamics. (a) Results are obtained at various values of the bias s (b) The black curve is a fit through the final three data points using a least squares fitting method of the form $k_\infty + A/n_{\text{c}}$ where $k_\infty = 0.891$ (3s.f.) and $A = 50.1 \times 10^3$ (3s.f.)

$\log \lvert t_{\text{obs}} \rvert$	$L = 20$	$L = 50$	$L = 80$
1	2.4×10^{-1}	3.5×10^{-1}	3.6×10^{-1}
2	2.1×10^{-1}	3.3×10^{-1}	3.5×10^{-1}
3	7.0×10^{-2}	3.0×10^{-1}	3.4×10^{-1}
4	1.2×10^{-2}	7.9×10^{-2}	2.2×10^{-1}
5	1.4×10^{-3}	1.2×10^{-2}	3.5×10^{-2}
6	1.6×10^{-6}	2.6×10^{-3}	4.9×10^{-3}

Table 5.1: Relative error in activity at $sL^2 = 5$ when varying the number of units of time. These results were obtained with $n_{\text{c}} = 10^3$ ($L = 20$), $n_{\text{c}} = 10^4$ ($L = 50, 80$) clones.

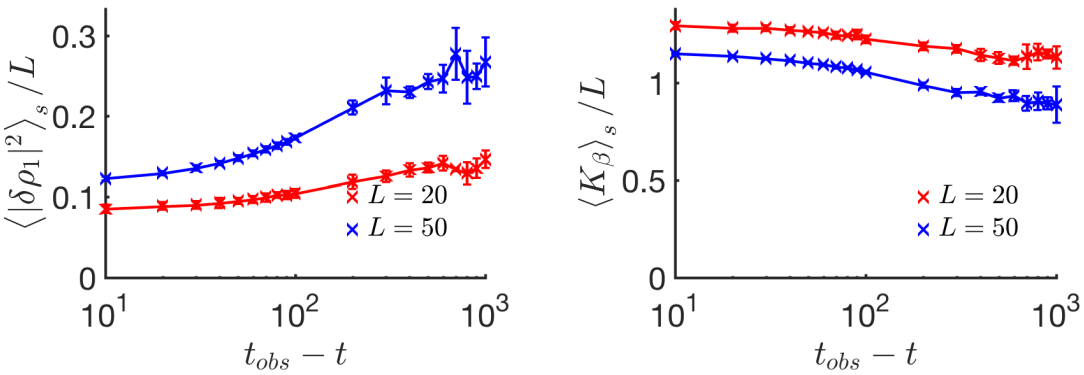
time as necessary to maximise efficiency.

To gain a better understanding of why the required number of units of time varies with system size as it does and why it takes the value that it does we measure how some observables evolve over time. In figure 5.17 we look at how the p_{ave} measure of two observables varies with time. In figure 5.17a we see that the first Fourier component of density of systems of size $L = 20$ has a timescale of $\sim 10^3$ units of time. When we increase the system size to $L = 50$ the timescale becomes longer and $t_{\text{obs}} \sim 10^3$ is insufficient for it to converge to its true value. We observe the same timescales in figure 5.17b when we measure the activity per cloning interval. To allow the errors incurred by this timescale to become small we hence require $t_{\text{obs}} \sim 10^4$ units of time to obtain values for $L = 20$ sites. We require even more than this to obtain results for systems of size $L = 50, 80$.



(a) $L = 50$ (b) $L = 80$

Figure 5.16: Directly measured activity with various values of the number of units of time t_{obs}. Results obtained at $n_{\text{c}} = 10^4$, $sL^2 = 5$, $\Delta t = 10$ over 10 repeats (exc. $t_{\text{obs}} = 10^6$, 1 repeat) with the [eq] (equal spacing) clone selection method and non-modified dynamics. The black curves are $1/t_{\text{obs}}$ fits through the final three data points of the form $k_\infty + A/t_{\text{obs}}$. These values are $k_\infty = 4.6, 6.98$ (3s.f.) and $A = 3.95 \times 10^3, 1.95 \times 10^4$ (3s.f.) for $L = 50, 80$ respectively.



(a) Fourier Denisty $\delta\rho_q$ (b) Activity K_β

Figure 5.17: Directly measured observables varying with time at $sL^2 = 5$ for systems of size $L = 20$ ($t_{\text{obs}} = 10^4$, $n_{\text{c}} = 10^3$) and $L = 50$ ($t_{\text{obs}} = 10^5$, $n_{\text{c}} = 10^4$). These values are obtained at $\Delta t = 10$ averaged over 5 repeats with the [eq] (equal spacing) clone selection method.

$\mathcal{O}$	$L = 20$	$L = 50$
r	3.66×10^{-1}	8.99×10^{-1}
$\lvert\delta\rho_1\rvert^2$	4.58×10^{1}	6.69×10^{2}
K^β	5.02×10^{1}	1.19×10^{2}
Υ	4.1×10^{-3}	3.78×10^{-4}

Table 5.2: Normalisation coefficients $c_{\mathcal{O}}(t')$ of various observables at $sL^2 = 5$ for systems of size $L = 20, 50$. The algorithm is run for $t_{\text{obs}} = 10^4$ ($L = 20$), $t_{\text{obs}} = 10^5$ ($L = 50$) units of time on $n_{\text{c}} = 10^3$ ($L = 20$), $n_{\text{c}} = 10^4$ ($L = 50$) systems with cloning intervals of $\Delta t = 10$ units of time.

We also measure the autocorrelations with respect to the final time of several observables. We use the same definition (4.7) of autocorrelations as we have used in chapter 4 to investigate the performance of the algorithm when measuring large deviations in the SSEP. In figure 5.18 the evolution of these autocorrelations over time can be seen. Their normalisation coefficients $c_{\mathcal{O}}(t')$ are presented in table 5.2. As the cloning factor Υ is calculated directly from the activity per cloning interval K^β, the autocorrelations of these two observables align. Therefore the black curves for the activity per cloning interval K^β do not appear clearly in figure 5.18 as they are behind the blue curves for the cloning factor Υ. The timescale for the autocorrelations of these observables to decay to zero in systems of size $L = 20$ is of the order 10^3 units of time as can be seen in figure 5.18a. The time required to obtain results for systems of this size is therefore at least 10^4 units of time. In figure 5.18b when we increase the system size to $L = 50$, we find that the timescale for autocorrelations to decay to zero is longer and hence that more units of time are required to obtain results from the algorithm.

The timescales associated with Fredkin Processes are therefore longer than those associated with the SSEP. This is because in the SSEP, the size of the spectral gap goes like L^{-2} [21]. In the Fredkin Process, however, Movassagh [93] has found that the size of the spectral gap goes like L^{-p} where $p \geq 2$ and $p \leq 15/2$. This means that as the transient timescale τ goes like the inverse of the spectral gap [21], the longer timescales in Fredkin Chains are expected. The macroscopic fluctuation theory that Lecomte, Garrahan and van Wijland [85] use to calculate the large deviation function of the SSEP, see chapter 3, assumes diffusive spreading [10] so that a perturbation in the density profile would diffuse like $t^{1/2}$. In the Fredkin Process, however, the Dyck Path must remain positive which induces long range correlations. These are what give the Fredkin Process its long timescales.

Figures 5.19 (a)-(b) include fits of the form $k_\infty + A/n_{\text{c}}$ as in equation (4.9) through the final three data points in each case as we have done for the SSEP

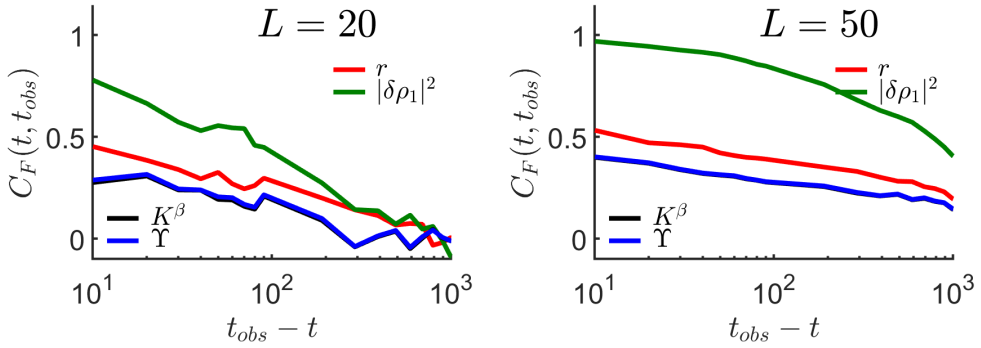


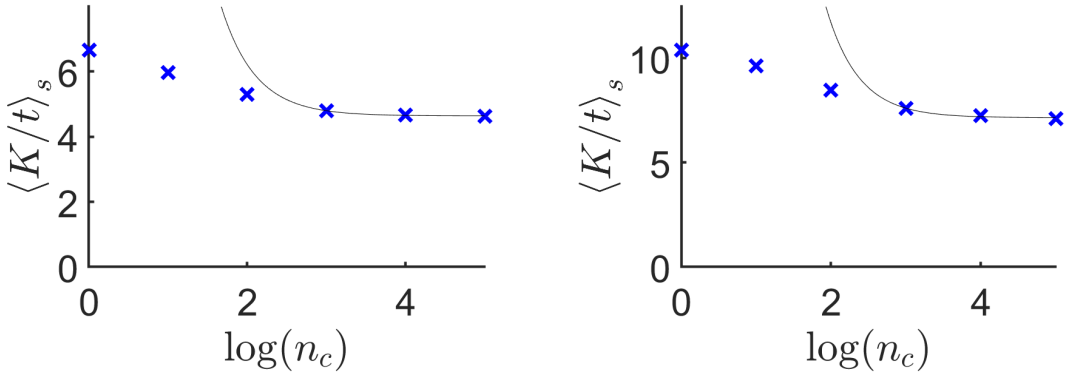
Figure 5.18: Directly measured autocorrelations with respect to the final time t_{obs} at $sL^2 = 5$ for systems of size $L = 20$ ($t_{\text{obs}} = 10^4$, $n_{\text{c}} = 10^3$) and $L = 50$ ($t_{\text{obs}} = 10^5$, $n_{\text{c}} = 10^4$). These values are obtained at $\Delta t = 10$ with the [eq] (equal spacing) clone selection method. The normalisation coefficients $c_{\mathcal{O}}(t')$ are presented in table 5.2.

$\log \lvert n_{\text{c}} \rvert$	$L = 20$	$L = 50$	$L = 80$
0	2.1×10^{-1}	3.0×10^{-1}	3.3×10^{-1}
1	1.0×10^{-1}	2.2×10^{-1}	2.7×10^{-1}
2	2.8×10^{-2}	1.2×10^{-1}	1.6×10^{-1}
3	7.0×10^{-3}	2.9×10^{-2}	5.8×10^{-2}
4	3.1×10^{-4}	2.2×10^{-3}	1.1×10^{-2}
5	9.9×10^{-4}	1.0×10^{-3}	3.7×10^{-3}

Table 5.3: Relative error in activity at $sL^2 = 5$ when varying the number of clones. These results were obtained with $t_{\text{obs}} = 10^4$ ($L = 20$), $t_{\text{obs}} = 10^5$ ($L = 50$), $t_{\text{obs}} = 10^6$ ($L = 80$) units of time.

in chapter 4. These fits provide us with an estimate of the value of activity as $n_{\text{c}} \to \infty$ as discussed. The values in table 5.3 then provide us with an estimate of the systematic error between the algorithm's value and this infinite value when we use a finite number of units of time. As in our paper [19] we assert a 2% criterion of the maximum error between the results from the algorithm and the results from the curve fit. This allows us to derive a value for the number of clones to use when obtaining results around the maximum susceptibilty. As always we look to use as few clones as necessary to maximise efficiency.

We obtain in table 5.3 and figure 5.19 that 10^3 clones are required for $L = 20$ sites for the algorithm to converge, when biasing activity in Fredkin Processes and 10^4 clones are required for $L = 50, 80$ sites. This is fewer than the number of clones required for convergence for the same system sizes when measuring the large deviations in activity in the SSEP. As discussed in chapter 4 the p_{ave} distribution which is defined in section 2.5 in definition (2.28) is obtained by a form of importance sam-



(a) $L = 50$ (b) $L = 80$

Figure 5.19: Directly measured activity with various values of the number of clones n_c. Results obtained at $t_{\text{obs}} = 10^5$, $sL^2 = 5$, $\Delta t = 10$ over 10 repeats (exc. $t_{\text{obs}} = 10^6$, 1 repeat) with the [eq] (equal spacing) clone selection method and non-modified dynamics. The black curves are $1/n_c$ fits through the final three data points of the form $k_\infty + A/n_c$. These values are $k_\infty = 4.65, 7.1$ (3 s.f.) and $A = 141, 499$ (3 s.f.) for $L = 50, 80$ respectively.

pling from the p_{end} distribution which is directly sampled by the cloning algorithm and is also defined in section 2.5 in definition (2.29). A lower bound on the number of systems that are required to obtain results from the algorithm is that there are enough systems that the p_{ave} distribution is sufficiently sampled. In figure 5.20 we make Gaussian fits to the values of p_{end} and p_{ave} that we sample. These Gaussian fits overlap much more than the equivalent fits for the SSEP in figure 4.7 which implies that less clones are required for convergence. This is in agreement with our results in table 5.3 and figure 5.19.

One reason that convergence with respect to the number of clones for Fredkin Processes requires less clones than for the SSEP may be that there are fewer configurations available for the system to exist in. This is due to the fact that there must be more occupied sites than unoccupied sites when reading from left to right across the lattice. Another reason may be that the low activity in the rare configurations is higher relative to the average activity than in the SSEP. This is because the activity is already restricted when there is no bias, this is also due to the rule that enforces that there must be more occupied sites than unoccupied sites when reading from left to right across the lattice. This theory is strengthened by the data for $L = 20$ and $L = 50$ in figure 5.20 which shows that the peaks in the p_{ave} and p_{end} distributions are close to one another.

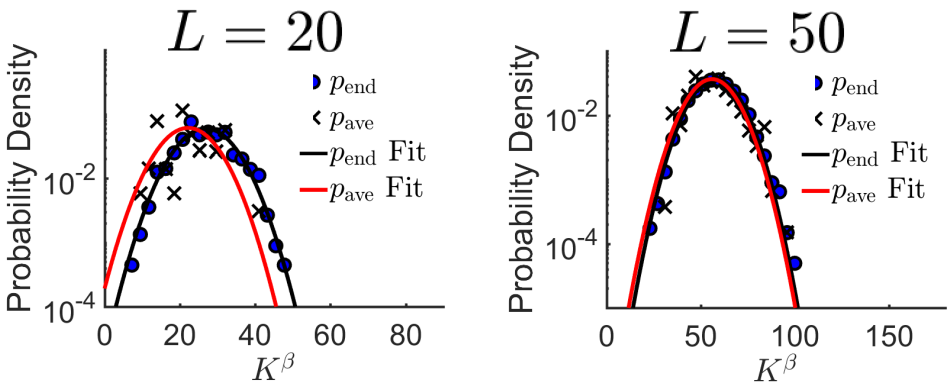
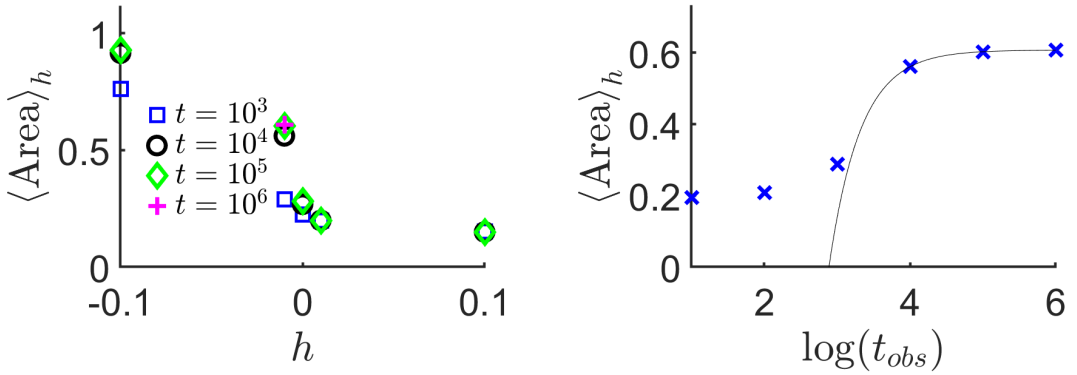


Figure 5.20: p_{ave} (crosses) and p_{end} (circles) distributions of activity per cloning interval K^β. The algorithm is run for $t_{\text{obs}} = 10^4$ ($L = 20$), 10^5 ($L = 50$) units of time on $n_c = 10^3$ ($L = 20$), $n_c = 10^4$ ($L = 50$) systems with cloning intervals of $\Delta t = 10$ units of time. The distributions are measured at $t = 9700$ (the 970^{th} cloning interval) at a bias $sL^2 = 50$.

5.4.2 Measuring Area Beneath the Dyck Path

We investigate time convergence when biasing the area by fitting a curve through the data of the form $1/t_{\text{obs}}$. As previously, we obtain an estimate of k_∞ as $t_{\text{obs}} \to \infty$ and of the fitting parameter A. We use a 2% criterion again to determine how many units of time are required for convergence at a fixed value of the number of clones n_c. In figure 5.21a we show convergence with respect to t_{obs} at various values of the bias h. Unlike when biasing the activity the steepest derivative in the observable that we are biasing exists at a negative bias. This implies that the variance in the area beneath the Dyck Path (the susceptibility) across the population is largest at a negative bias so we expect time convergence to be hardest when $h < 0$. This is seen at $h = -0.01$ where $t_{\text{obs}} = 10^5$ units of time are required for convergence when $L = 50$ as opposed to the other values of h which we require $t_{\text{obs}} = 10^4$ units of time to converge.

In figure 5.21, we show the convergence with respect to time t_{obs} at $h = -0.01$. As with the results obtained when we biased activity in the SSEP (section 4.3) and the Fredkin Process (section 5.4.1) we fit a curve through the data of the form in equation (4.6). Similarly to when we bias the activity in the Fredkin Process, when we bias the area in the Fredkin Process $t_{\text{obs}} = 10^5$ units of time are required for convergence when $L = 50$, as shown in table 5.4. When we bias to the maximum susceptibility we bias to the trajectories with the longest timescales. This means than when we bias the activity to the maximum susceptibility we bias to the same trajectories as when we bias the area beneath the Dyck Path to the maximum



(a) Various h (b) $h = -0.01$

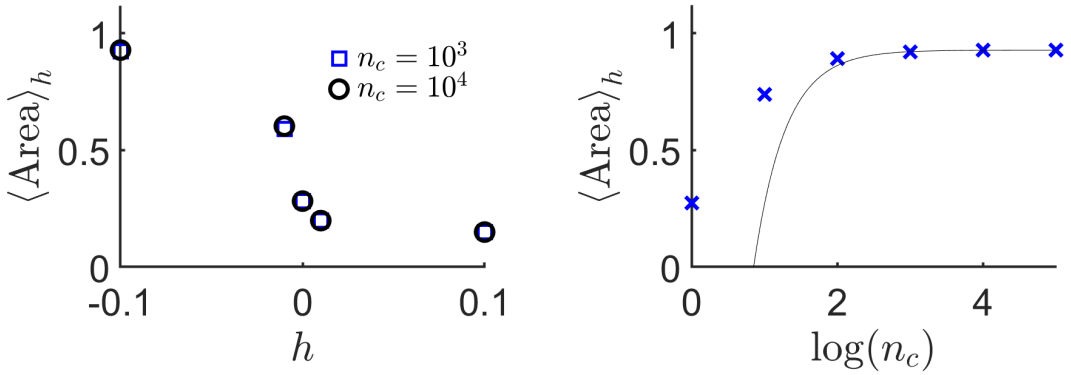
Figure 5.21: Time convergence of the directly measured area under the bias h. Results obtained at $n_c = 10^4$, $L = 50$, $\Delta t = 10$ over 10 repeats with the [eq] (equal spacing) clone selection method. (a) Results are obtained at various values of the bias h (b) The black curve is a fit through the final three data points using a least squares fitting method of the form $k_\infty + A/t_{\text{obs}}$ where $k_\infty = 0.607$ (3s.f.) and $A = -468$ (3s.f.).

$\log \lvert t_{\text{obs}} \rvert$	$L = 20$	$L = 50$
1	4.8×10^{-1}	2.1×10^{0}
2	3.3×10^{-1}	1.9×10^{0}
3	6.6×10^{-2}	1.1×10^{0}
4	6.7×10^{-3}	8.3×10^{-2}
5	1.1×10^{-3}	8.4×10^{-3}
6	2.9×10^{-4}	1.6×10^{-4}

Table 5.4: Relative error in area at $h = -0.01$ when varying the number of units of time. These results were obtained with $n_c = 10^4$ ($L = 50$) systems.

susceptibility. This explains why they require the same number of units of time to converge.

We use the same curving fitting techniques as used in subsections 4.3.2 and 5.4.1. We fit a curve of the form $k_\infty + A/n_c$ through our data as in equation (4.9). Again we use a 2% convergence criterion to determine how many clones are required to converge the results at a fixed value of the number of units of time. In figure 5.22 we show the convergence with respect to the number of clones n_c for systems of size $L = 50$. We find that the number of clones required for convergence are $n_c = 10^2$ and $n_c = 10^3$ systems for systems of size $L = 20$ and $L = 50$ respectively as shown in table 5.5. The values of the the relative error that are obtained when biasing the area beneath the Dyck Path are similar to those obtained when biasing the activity. This is likely to be because the p_{ave} and p_{end} distributions of area when biasing area have a similar amount of overlap to the p_{ave} and p_{end} distributions of activity when



(a) Various h (b) $h = -0.1$

Figure 5.22: Clone convergence of the directly measured area under the bias h. Results obtained at $t_{\text{obs}} = 10^5$, $L = 50$, $\Delta t = 10$ over 10 repeats with the [eq] (equal spacing) clone selection method. (a) Results are obtained at various values of the bias h (b) The black curve is a fit through the final three data points using a least squares fitting method of the form $k_\infty + A/n_c$ where $k_\infty = 0.263$ (3 s.f.) and $A = 3.07 \times 10^{-2}$ (3 s.f.).

biasing activity at the values of the bias that we have considered.

If we were to bias one quantity and measure another we would expect to require as many clones for the values to converge as when we measure the quantity that we are biasing. This is because we would require that we access the same rare trajectories as when we measure the quantity that we are biasing. Furthermore, our results in figures 4.4 and 4.5 for the SSEP and in figures 5.17 and 5.18 for the Fredkin Process show that the timescales of different quantities under the same dynamics are the same. This means that the number of units of time required for the errors incurred by these time scales to become small is similar when we bias one quantity and measure another to when we measure the quantity that we are biasing. This does, however, depend on the relative of the value of the quantity that we are measuring at the final time to its value in the TTI regime. Therefore, when we bias one quantity and measure another, the number of units of time required for its convergence depends on its relative value at these two times.

5.5 Fredkin Summary

We have obtained results when biasing activity in the Fredkin Process. We have then verified the accuracy of our results by comparison with some theoretical values of the large deviation function for a small system and theoretical area beneath the Dyck Path at no bias obtained from Brownian Bridges. In the range $s = -0.1$ to

$\log \lvert n_c \rvert$	$L = 20$	$L = 50$
0	1.1×10^{0}	2.4×10^{0}
1	1.8×10^{-1}	2.6×10^{-1}
2	1.6×10^{-2}	4.1×10^{-2}
3	5.7×10^{-4}	7.2×10^{-3}
4	4.4×10^{-4}	9.9×10^{-4}
5	4.5×10^{-4}	1.8×10^{-4}

Table 5.5: Relative error in area at $h = -0.1$ when varying the number of units of time. These results were obtained with $t_{\text{obs}} = 10^5$.

0.1 we have obtained the large deviation function and directly measured activity from the algorithm. Across this range of biases we have measured the clustering of particles by measuring density profiles, the directly measured area beneath the Dyck Path, the centre of mass and typical trajectories. When s is negative we see that the particles tend to be as diffuse as possible and consequently have a centre of mass near the centre of the lattice and that the area beneath the Dyck Path is small. As we transition to a positive bias the particles cluster specifically on the left hand end of the lattice, the sites with small indices. As a consequence the centre of mass tends towards its minimum possible value 0.25 and the area beneath the Dyck Path tends towards its maximum possible value 1.

We have then investigated the activity, large deviation function and susceptibility as a function of the sL^2 scaling of the bias that we made when analysing the phase transition in the SSEP in chapter 3. As with the SSEP we have found that the values of activity and the large deviation function agree at this scaling. We have found that the position of the peak in susceptibility as $L \to \infty$ is at a far smaller value of the bias sL^2 than the position of the peak in the SSEP. Like the SSEP, as the system size increase the value of the bias sL^2 at which the peak occurs decreases and in the Fredkin Process it may decrease to 0.

We have then investigated large deviations in the area beneath the Dyck Path. We have done this by obtaining results as a function of the bias h and found that the properties of the trajectories transition in the opposite way to the properties of the trajectories under the bias s. At a positive bias the particles are diffuse and the centre of mass approaches 0.5, its maximum allowed value. There is a sharp transition around $h = 0$ that is seen in the large deviation function and directly measured area. At a negative bias we see the particles clustering to the left hand boundary of the lattice on the sites with the smallest indices. The larger value in the area beneath the Dyck Path at negative h corresponds to a lower value in the centre of mass that tends towards its minimum allowed value 0.25.

We have measured the number of clones and the amount of time required for the algorithm to converge when obtaining results for the Fredkin Process. We have found that it requires fewer clones than the SSEP but more units of time. The reason that more time is required is that the Fredkin Process has longer time scales associated with it because of long range correlations. The smaller number of clones required by the Fredkin Process for convergence may be caused by the fact that there are fewer configurations available for the system to exist in. It may also be that the particles in rare configurations hop at a higher rate relative to the average rate of the process than is the case in the SSEP.

Chapter 6

Computational Approaches for Measuring and Improving Speed and Efficiency

We have shown that our implementation of the algorithm produces useful results in previous chapters. We now investigate the quickest way to computationally generate these results. Researchers that use the cloning algorithm to collect results seek to do so in a way that is quick but also in a way that is not expensive in the amount of resources that it uses. This means that our objectives are to obtain large speeds but also large efficiencies. One way to do this is to use computational parallelism so that multiple processors execute the algorithm simultaneously.

Here we state the effect of various parallelisation techniques on the speed and efficiency of the code. Also we show the effectiveness of using MPI techniques in terms of the effect that it has on the codes speed and whether it is worthwhile using them when obtaining results from the algorithm. We specifically focus on the run time of the code when simulating the SSEP and a particular test case of typical parameters used to obtain results in chapters 3, 4.

We measure the communications patterns of typical MPI implementations in terms of the number of communications sent between pairs of processors and investigate ways of reducing the number of communications sent between them. We also consider how our implementations scale across multiple processors and what options to use during compilation such as the compiler and optimiser.

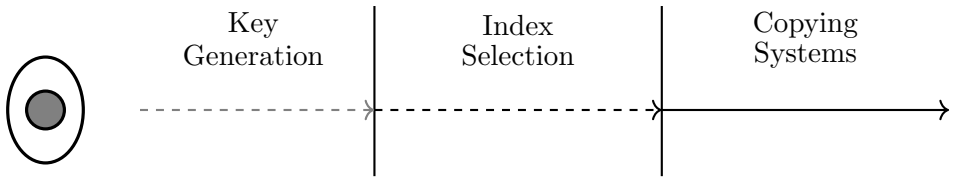


Figure 6.1: Serial implementation of the algorithm code run on one node (ellipse) and one processor (grey circle).

Process	Modified SSEP
Observable	Activity
Number of Clones	$n_{\mathrm{c}} = 10^5$
Number of Units of time	$t_{\mathrm{obs}} = 10^4$
System Size	$L = 50$
Number of Particles	$N = 25$
Cloning Interval	$\Delta t = 10$

Table 6.1: Set of Parameters for a Test Case at which we Obtain Results.

6.1 Serial Code and Definitions

To implement the algorithm we have written a *C++* code which implements each stage of the cloning algorithm. The dynamics stage is run on all n_{c} systems in a for loop. These are then used to create a binary search key which is used to select the indices of the systems to clone. Finally, the cloning stage is completed using a *copy* function to clone the systems. A diagram representing the clone selection and cloning stages in the serial implementation is shown in figure 6.1. We have written several *C++* implementations of the algorithm to investigate which computational techniques maximise the speed and efficiency at which we can obtain results from the algorithm.

To test and compare the run times of each implementation we have defined a set of parameters at which to obtain results as a test case (see table 6.1). We generally obtain results for this test case at a bias $sL^2 = 20$ using the [eq] clone selection method. This serial code takes 5515s to obtain results on one processor on one node when averaged over 20 runs. We define the speed-up $\mathcal{S}_m$ of implementation m as

$$\mathcal{S}_m = \mathrm{RT}_s/\mathrm{RT}_m, \tag{6.1}$$

where RT_m is the run time of implementation m and RT_s is the run time of the serial code. The corresponding efficiency $\mathcal{E}_m$ of implementation m is defined as

$$\mathcal{E}_m = \mathcal{S}_m/n_t, \tag{6.2}$$

where n_t is the number of threads that the code is running on. In general, both the speed-up $\mathcal{S}_m$ and the efficiency $\mathcal{E}_m$ are dependent on the number of threads n_t. All of our results were obtained on (8-core) Intel Xeon E5-2650V2 Ivybridge processors at 2.6 GHz. The computer nodes on which the implementations are run each have a maximum of 16 threads that can be used simultaneously with shared memory.

6.2 OpenMP Code

We parallelise the code using a parallelisation technique called OpenMP [93, 25]. This means that multiple processing cores are used to run the computer code simultaneously. We expect this to decrease the run time of the code. Under OpenMP multiple threads are used so that each thread performs computations independently of the other threads. They do this with shared memory so that each thread has access to the same memory as all of the other threads. OpenMP may only be run on at most one node at a time.

The OpenMP code parallelises by using 16 threads on one node. The systems are shared across all threads by shared memory and so the cloning communication stage is implemented by the *C++ copy* function which performs a memory copy. The dynamics are parallelised so that each thread implements the dynamics on an equal share of the systems. The binary search key is generated by one thread and available to all others through shared memory. The systems are indexed $0 \ldots n_{\mathrm{c}} - 1$ and each selects which system will be copied to replace it in the next set of systems for the next cloning interval. This clone selection process is parallelised and the indices of the systems to be cloned are known by all other threads through shared memory.

The cloning is also parallelised so that each thread clones an equal share of the number of systems to be copied. A diagram of the clone selection and cloning stages in the OpenMP implementation are shown in figure 6.2. The run time for the OpenMP code is 349s when run on 16 threads on one node across 10 repeats. This corresponds to a speed-up of 15.8 and an efficiency of 98.8%. This is a good speed up and means that a set of results that would have taken two weeks to obtain with the serial code can be obtained with the OpenMP code in less than a day. The efficiency is very high and indicates that the additional run time generated by the OpenMP overheads is small relative to the overall run time of the code.

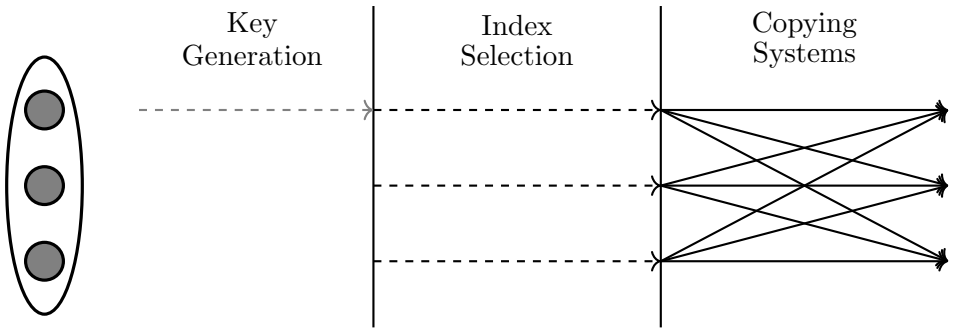


Figure 6.2: OpenMP implementation of the algorithm code run on one node (ellipse) and multiple threads (grey circles).

6.3 MPI Code

We also use a parallelised MPI [54, 108] code. Multiple processing cores are again used to run the computer code so that each processor executes code independently. Under MPI there is not shared memory so MPI communications are required to send information from one processor to another. One advantage that MPI has over OpenMP is that it is not restricted to only being run on one node. This means that it should be possible to obtain much larger speed-ups. It may then be possible to attain results for large systems in a short period of time by using MPI implementations on a large number of nodes.

This code is parallelised so that we have 16 processors per node and the ability to run the code across multiple nodes. The dynamics are run such that an equal share of the systems are run on each processor. Each processor goes through its share of systems and runs the dynamics on them one at a time and calculates their associated cloning weight. The cloning weights are then communicated to all other processors via *MPI_Allreduce*. Each processor then independently generates an identical binary search key. This is so that one processor does not have to do all the work of clone selection and so that MPI overheads are not incurred by communicating a binary search key between processors. The systems then each choose which system will replace them in the next set of systems. This information is then communicated to all of the other processors using the *MPI_AllGather* function. Systems are communicated through MPI functions sending systems between processors. A diagram of the clone selection and cloning stages of the algorithm is shown in figure 6.3.

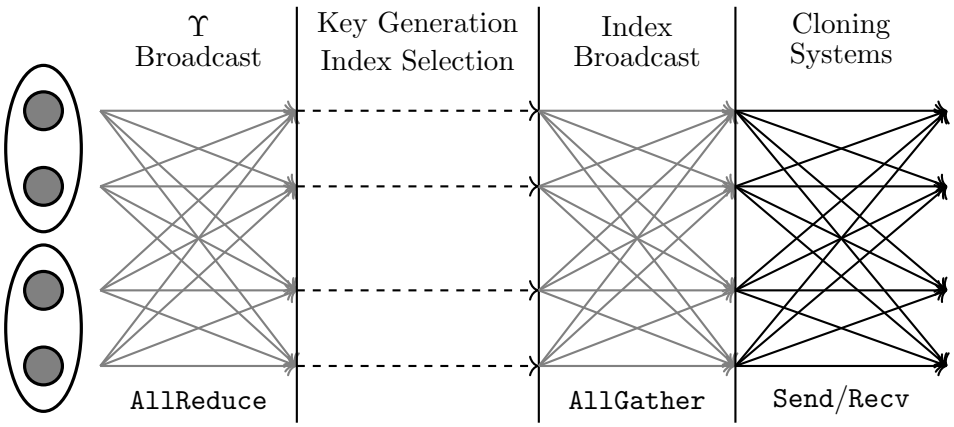


Figure 6.3: MPI implementation of the algorithm code run on two nodes (ellipses) and multiple processors (grey circles). Arrows are MPI communications, black arrows are pointwise and grey arrows are collective.

6.3.1 MPI *Pack* Functions

Generally systems are in a state defined by several components. In our test case the SSEP involves a one-dimensional lattice and hopping particles. The components that therefore define the state of the system are site occupancies, particle positions, particle freedoms and the escape rate. So far when we have been performing the cloning process, each of the components has been copied separately. In the MPI implementation this corresponds to one MPI message per component.

In principle, MPI implementations increase in speed as the message size increases if the number of MPI messages is decreased proportionately. This means that when sending systems via MPI it is expected that packing a system's components into one MPI message increases efficiency. We pack/unpack components using the MPI functions MPI_Pack/MPI_Unpack. The speed of the implementations may be reduced by the additional computation required to execute these functions. When we pack system components into one message Table 6.2 shows that this implementation [*Components*] is quicker than the MPI implementation [*None*] that does not pack components.

Furthermore, it is possible to pack multiple systems from a processor into one MPI message. This reduction in the number of MPI messages is expected to increase the code's speed further. Table 6.2 shows that the implementation [*Systems*] which packs the components of multiple systems into each array and then sends it is quicker than the implementation [*Components*] in which the components of only one system are packed into each array to be sent.

Amount of Packing	Speed-Up	Efficiency (%)
None	48.6	75.9
Components	49.8	77.8
Systems	55.3	86.4

Table 6.2: Run times obtained by MPI implementations with different amounts of packing when run on 4 nodes and 16 processors per node. The results are obtained at the set of parameters in the test case (see table 6.1) and at a bias $sL^2 = 20$.

6.4 Reduced Communications

Each pair of processors send systems to each other under the simple set of communications. If processor A and processor B are sending systems to one another this means that there is an inefficiency because the MPI communications required to send systems to one another could be replaced by each one of these processors copying as many systems as possible internally. We implement this using a pairwise cancellation procedure and so for each pair of processors systems are only sent in one direction. All of the other systems that would have been sent are copied internally within the processor they exist on using the *C++ copy* function to perform a memory copy.

We define $\Gamma_{A,B}$ as the number of systems sent from processor A to processor B. Under this communication reduction the number of systems $\Gamma^*_{A,B}$ sent from processor A to processor B is

$$\Gamma^*_{A,B} = \Gamma_{A,B} - \min(\Gamma_{B,A}, \Gamma_{A,B}), \tag{6.3}$$

where $A \neq B$. This significantly decreases the number of messages sent by MPI, see section 6.4.1 below. Furthermore each MPI message contains fewer systems which leads to smaller MPI messages which in principle are quicker to send.

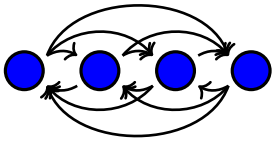
To assert that each system is copied the number of times determined by the cloning process and that the number of systems on each processor remains fixed, the decrease in MPI communication is offset by increased internal copying within a processor. The number of systems copied internally $\Gamma_{A,A}$ within each processor A using a *copy* function is increased to

$$\Gamma^*_{A,A} = \Gamma_{A,A} + \sum_{B \neq A} \min(\Gamma_{B,A}, \Gamma_{A,B}), \tag{6.4}$$

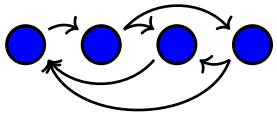
where the sum is over all other processors. As table 6.3 shows this approach increases the speed of the code. Our main reason for producing MPI implementations is to obtain higher speed-ups than we did with our OpenMP code. We can achieve this

Communications Patterns	Speed-Up	Efficiency (%)
Simple Comms	54.6	85
Reduced Comms	59	92

Table 6.3: Run times obtained by MPI implementations with and without reduced communications when packing systems and running the code on 4 nodes and 16 processors per node. The results are obtained at the set of parameters in the test case (see table 6.1) and at a bias $sL^2 = 50$.



Simple Communications



Reduced Communications

Figure 6.4: An example of reduced MPI communication between four processors (blue circles). Communications are reduced by point-wise cancellation so that communication only occurs in one direction between each pair pf processors.

as MPI implementations can be run on multiple nodes whereas OpenMP codes can only be run on one node. Our focus here is not on the resources spent on obtaining these higher speed ups and so the fact that under reduced communications the MPI implementation remains less efficient than the OpenMP code is less important than the fact that it achieves larger speed-ups.

6.4.1 Communication Patterns

The diagram in figure 6.4 shows a typical reduction in communications between four nodes under our communications reduction procedure. The communications patterns in figure 6.5 show the number of systems sent between each pair of processors that we measure when we run the algorithm code. The reduction in the number of communications is clearly visible with messages only being sent in one direction for each pair of processors. This communication reduction hence reduces the number of messages by a factor of at least 2 and the average message size from typically containing around 25 systems to around 5 systems. This corresponds to the total number of systems sent by each processor reducing by about a factor of 10.

The reduction in the number of systems sent by MPI by a factor of 10 corresponds to an increase in the speed-up of the code of less than 10%, see table 6.3. This is due to the fact that the reduced communications only increase the speed

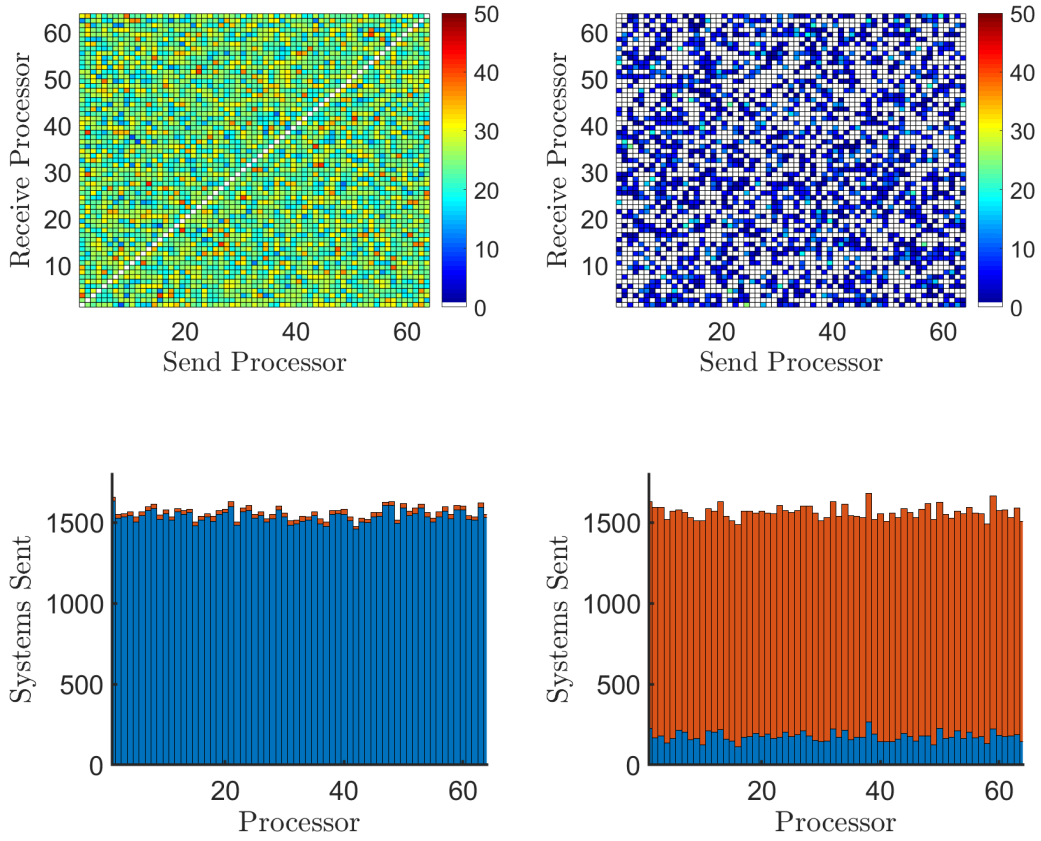


Figure 6.5: The number of systems sent between each pair of processors using MPI when evolving systems at the set of parameters in the test case (see table 6.1) and at a bias $\lambda = 20$. Simple communications (*left*) and reduced communications (*right*) using the independent [iid] clone selection method. These communications are in the 500^{th} cloning interval ($t = 5000$). *Bottom Row:* The number of systems sent by each processor using MPI (blue) and those whose states are simply copied (orange).

of the communications stage of the algorithm, not the dynamics stage. Another reason for this is that the increase in speed is largely only due to the reduction in the number of messages, which is a factor of 2, not the size of the messages.

6.5 Non-Blocking Implementations

When using MPI to parallelise our code we use send and receive functions to perform MPI communications. When these communications are *blocking* [39] the processor that executes the send function waits until the message has been sent and can be overwritten before it executes any more code. Similarly, the processor that executes the receive function waits until the message has been received before it executes

any more code. Conversely *non-blocking* MPI communications allow computation to occur whilst messages are being transmitted.

When non-blocking communications are performed and the receive function is posted it returns a *request* object [107]. After the receive function is posted at some point the receiving processor will require that the MPI message has been received before performing some particular computation. To ensure that this has happened a *wait* function can be called to wait for the MPI message to be received [104]. This *wait* function can wait on one or multiple MPI messages [72]. When the *wait* function waits on multiple MPI messages it contains an MPI request array of requests [30]. When a communication completes this is when the *request* object is returned by the *wait* function.

6.5.1 Test Codes

One drawback of our previous implementations is that they are blocking. The consequence of this is that dynamics and communication cannot happen simultaneously. Furthermore, some nodes sit idle during the communications stage whilst other nodes communicate systems to one another. A solution to this is to use non-blocking implementations and non-blocking MPI functions. This means that computations such as the dynamics can be being performed whilst communication (i.e. the cloning process) continues to be executed in the background. To test the efficiency of non-blocking MPI codes compared to blocking MPI codes we produce some simple test codes that work at a fixed number of systems per processor. We have done this as immediately creating a non-blocking version of our algorithm code may be unnecessary if non-blocking communications do not work effectively for the system sizes and communication patterns that we are considering.

Test Code A

In this simple test code we approximate the dynamics computation by the production of random numbers. These random numbers are used to fill one integer array of length 50, two integer arrays of length 25 and a double. In this test code, the replacement systems are randomly chosen for the copying process. An *MPI_Allreduce* function is used to communicate between all processors which systems are to be copied. The same function is used to communicate between all processors which processor each system is to be copied from. There are four MPI messages sent for every system that is copied, one for each component.

For this test code we have two implementations, one is blocking and the other

Implementation	Run Time(s)			
	$nc_t = 10^2$		$nc_t = 10^3$	
	$t = 10^3$	$t = 10^4$	$t = 10^3$	$t = 10^4$
Non-Blocking (WaitAny)	24.6	239	262	2560
Blocking	12.7	128	125	1250

Table 6.4: Run time of test code A which copies four components using random communication patterns and mimics the dynamics process by the production of random numbers. Results obtained on 4 nodes of 16 processors per node and there is one dynamics stage and one communications stage per unit of time.

is non-blocking. The blocking implementation uses *MPI_Recv* and *_Send* functions and the non-blocking code uses the *MPI_Waitany* function to wait for any system. Our results in table 6.4 show that both cases scale approximately linearly with time. Furthermore, they show for both values of clones per thread nc_t, that the blocking is quicker than the non-blocking code. This suggests that the blocking communications are quicker than the non-blocking communications. This could either be due to the *MPI_Recv* and *_Send* functions being quicker than the *MPI_Waitany* function or the non-blocking messages taking a longer time to send than the blocking messages. In both cases this may be due to the number of systems that we are trying to send during the communications process.

Test Code B

In this test code we use a *nanosleep* function to replicate the time spent by the algorithm code computing the dynamics stage. The communications stage is replicated by a fixed communications pattern where systems are sent between processors using an MPI function and within a processor using a *copy* function. This can be seen in figure 6.6 where each communication represents four MPI messages, one for each component. No random numbers are used in either the dynamics or communications stages of these test codes. When the test code is run for a given number of communication stages it is trivial to calculate what the values of the array elements are expected to be and hence to perform a consistency check that the test codes are functioning as expected.

To test how effective that the non-blocking functions are we compare four implementations of the test code. One of these implementations is blocking and three are non-blocking. As with the previous test code we have an *MPI_Waitany* implementation. In addition, for this test code we have *MPI_Wait* and *MPI_Waitsome* implementations. The *MPI_Wait* function has one MPI request and waits for one particular system to be received. The *MPI_Waitsome* function has an MPI request

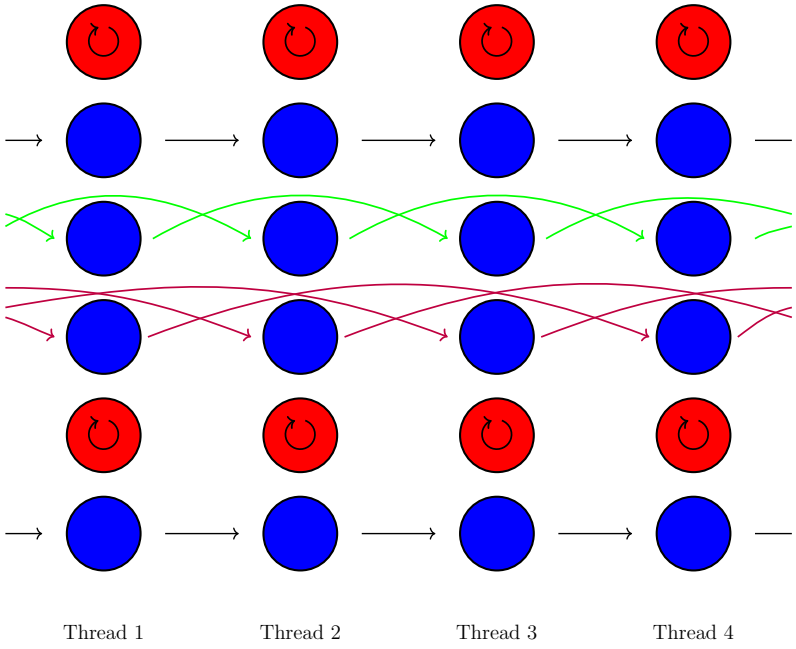


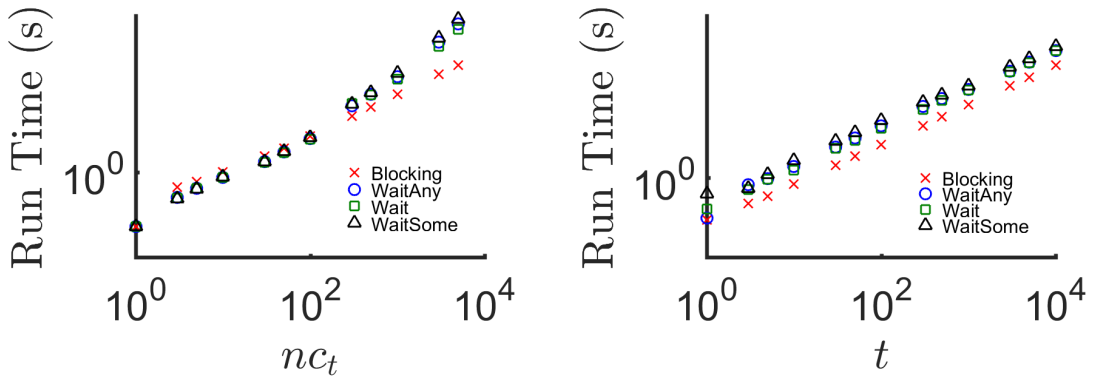
Figure 6.6: Communication between systems and threads in test codes with fixed communications patterns. Each circle represents a system. Red systems are copied within the same thread using a *copy* function and blue systems are sent to other threads using MPI functions. Black arrows copy to the next processor, green arrows to two processors away and purple arrows to three processors away.

array for all of the systems. It waits for at least one system to have been received and once it has the indices of all of the systems to have been received are returned.

As figure 6.7 demonstrates, the three non-blocking implementations are a similar speed to one another regardless of the number of units of time or clones per processor that the test code is run for. With a small number of clones per processor (< 200) the non-blocking codes are quicker than the blocking code as figure 6.7 shows. A better approximation of the parameters used when we obtain results from the algorithm code employs a lot more clones per processor than this and in this regime the blocking code is faster.

Array-100 Codes

We have an alternative set of test codes called the Array-100 codes. Again, we use the communications patterns of test code B as in figure 6.6. Also like test code B the dynamics are mimicked by the *nanosleep* function. In this case we have one integer array of length 100, which is again the equivalent number of array elements to a lattice of length $L = 50$. We again have four implementations of the array-100 code, three are non-blocking and use the *MPI_Isend* and *_Irecv* functions and one



(a) $t = 10^3$ (b) $nc_t = 10^3$

Figure 6.7: Run time of test code B which copies four components using a fixed communications pattern and mimics the dynamics process using a *nanosleep* function. Results are obtained on 4 nodes of 16 processors per node and there is one dynamics stage and one communications stage per unit of time.

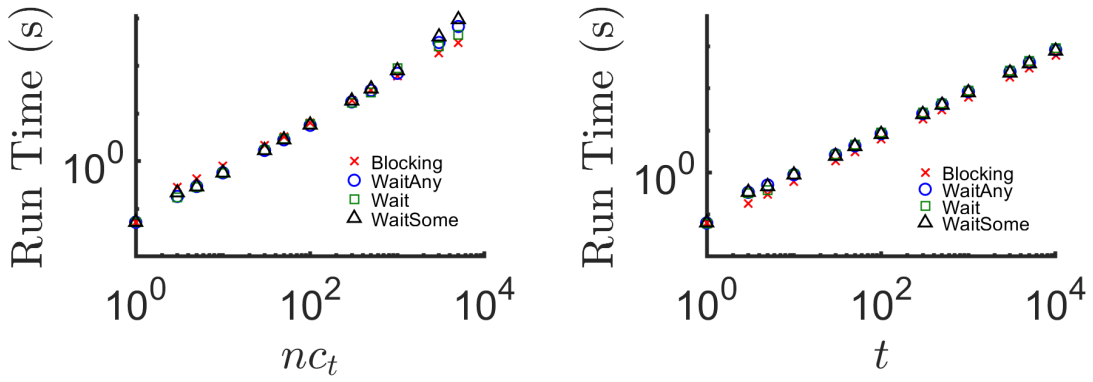
is blocking.

Figure 6.8 shows that when there are $nc_t < 2000$ clones per processor, the blocking code is slower than at least one of the non-blocking codes. When we are obtaining results from our algorithm code we often need more clones than this corresponds to and hence we are looking at a regime where the blocking code is quicker. This is a common observation across all of the non-blocking codes that we have considered. When we have a large number of systems and hence communications and MPI requests, the code takes a long time to run.

Balaji et al [5] have investigated why the run time of multi-request MPI functions increases so quickly with its number of internal requests. They have found that the run time of these functions scales linearly with the number of MPI requests passed to them because of the time they take to gather the requests together. The fact that the number of times that the *MPI_WaitAny* and *_WaitSome* functions are called and the length of time that it takes for them to be called both scale linearly with nc_t explains why the run times of the codes scale so quickly with the number of clones per processor.

Two Request Arrays

A solution to the problem of long request arrays that take a long time to computationally gather is to split the request array into two smaller arrays. The corresponding wait function then has to collect together only half as many requests each time it is called. This increases the speeds of the wait function each time that it is called and hence the code overall. In table 6.5 we see that increasing the number of



(a) $t = 10^3$ (b) $nc_t = 10^3$

Figure 6.8: Run time of the test codes which copy one component in the communications process, an integer array of length 100. The communications patterns are fixed as in figure 6.6. Results are obtained on 4 nodes of 16 processors per node. There is one dynamics stage and one communications stage per unit of time.

Implementation	Run Time(s)		
	$nc_t = 10^3$	$nc_t = 3 \times 10^3$	$nc_t = 5 \times 10^3$
WaitAny (One Array)	70	311	674
WaitAny (Two Arrays)	64.8	256	549
Blocking	62.3	186	310

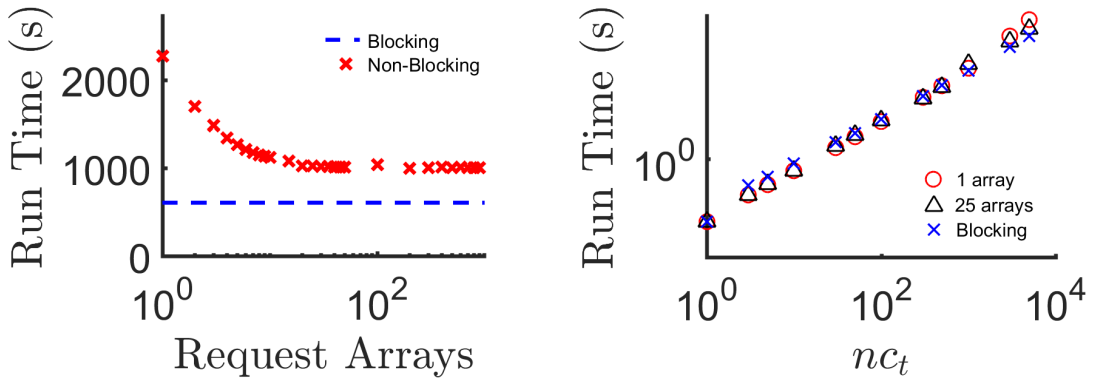
Table 6.5: Run times of the Array-100 codes when the request array of the WaitAny function is split. Compared with the case where the request array is not split and the blocking implementation at $t = 10^3$ units of time. There is one dynamics stage and one communications stage per unit of time.

WaitAny request arrays to two does decrease the run time of the code but that it is still slower than the blocking implementation.

Multiple Request Arrays

As splitting the request array make the non-blocking WaitAny code quicker we investigate how much the codes speed can be increased by further splitting the request array into equally sized smaller arrays. Every reduction in the size of the array corresponds to a reduction in the number of requests that have to be collected each time that a message is received and a wait function called.

In figure 6.9 we see that at a fixed value of clones per processor nc_t, the run time decreases as the number of request arrays increases up to a point where the run time plateaus and does not change as the number of request arrays is increased further. We see that the run time of the non-blocking code is never as quick as the run time of the blocking code. These results were at $nc_t = 10^4$ which corresponds



(a) $t = 10^3$, $nc_t = 10^4$ (b) $t = 10^3$

Figure 6.9: Run time of the test code which uses multiple request arrays. The components being copied are of length 100 and the communications patterns are fixed as in figure 6.6. Results are obtained on 4 nodes of 16 processors and there is one dynamics stage and one communications stage per unit of time.

to more than $n_c = 6 \times 10^5$ systems in total.

Figure 6.9 also shows how the run time of this code changes as the number of clones per processor nc_t is varied. When we run the code with a large number of systems per thread $nc_t > 10^3$ we generally find that the blocking code is faster than the non-blocking code regardless of how many smaller segments that the request array is split into. We also find that when the request array is split into 25 smaller arrays that the non-blocking code is quicker than when there is one request array. When the number of clones per processor nc_t is less than 10^2 the non-blocking code exhibits more efficient run times than its blocking equivalent regardless of whether the request array is split.

6.5.2 Packing Multiple Systems

Having generated results for codes where systems and their components are packed together we now consider the case where we do this alongside using non-blocking functions. The main reason that the non-blocking codes take such a long time to run is that there are a large number of MPI communications and there is consequently a large number of MPI requests in the MPI request array. This is the case when there are many components (test codes A and B) and also when there is one long component (array-100 test codes) which is effectively the same as the case where several components have been packed.

The main advantage of a non-blocking implementation in which systems to be sent between each pair of processors are packed is that there are far fewer MPI communications. We define the abbreviation cpm as meaning clones per message.

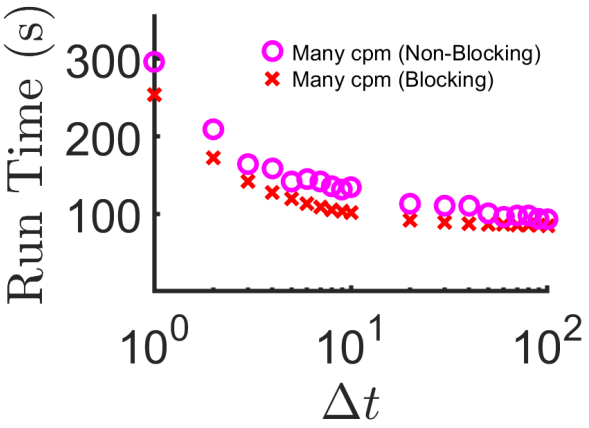


Figure 6.10: Run time of a blocking code with system-packing and a non-blocking code with system-packing. Results obtained on 4 nodes of 16 processors per node and averaged over 10 repeats. Results are obtained at the set of parameters in the test case (see table 6.1) at a range of values of the cloning interval and at a bias $sL^2 = 20$.

In the case where we have $n_c = 10^5$ systems on 4 nodes as in figure 6.10 the packing of systems reduces the number of MPI communications from of the order $\sim 10^5$ to $\sim 4 \times 10^3$. We find that this non-blocking implementation is not as fast as its blocking equivalent as the results presented in figure 6.10 demonstrate. We are able to adjust the frequency of communication stages by scaling the size of the cloning interval Δt. We find that at all values of the cloning interval the blocking implementation is quicker. By comparison with the results for the test code with arrays of length 100 in figure 6.8 we see that for the number of clones per processor that we are using we would expect the blocking code to be quicker because of the effect of the quick increase in the time spent calling wait functions with the number of clones per processor.

6.6 Hybrid Implementations

The OpenMP code does not exhibit the communication inefficiencies observed in the MPI implementations. One way to deal with this is to have shared memory across a node and hence increase the amount of internal copying. This means that there is no use of MPI functions to perform copying between processors that are on the same node. In addition, we may use the packing function so that multiple systems that are on the same node but different processors may be sent together in one communication.

We have written a hybrid code that performs this packing routine of all sys-

tems to be sent between each pair of nodes. In figure 6.11 the diagram depicts the clone selection and cloning stage in the Hybrid implementation. In figure 6.12 we then present the run time of this implementation compared to other implementations with different amounts of packing. We plot these results against the size of the cloning interval Δt which is effectively a parameter which decides the frequency of communication. As figure 6.12 shows, the implementations have similar run times when there is a low amount of communication, at large values of Δt.

When the cloning intervals are small the communication is much more frequent. We know that the implementation that packs together systems on the same processor is more efficient than the implementation that only packs components. We see in this regime that packing more systems together decreases the codes speed in the hybrid implementation where all the systems sent between each pair of nodes are packed together. We observe this despite the large reduction in the number of MPI messages.

When we hybridise the OpenMP and MPI codes we use the typical procedure of using only one processor (rank 0) on each node to execute the MPI communications. The consequence of this is that the communications are performed by n_N processors where n_N is the number of nodes. This corresponds to a reduction in the number of processors communicating MPI messages by a factor of 16. As the blocking codes send one MPI message at a time this means there is a large increase in the amount of time that messages are waiting for the previous message to be sent before they are able to sent. This explains the increase in run time that we see in figure 6.12.

6.6.1 Test Code

When we run the hybrid code with all systems being sent from one node to another packed into one buffer we find that it is slower than the other blocking implementations with packing. The obvious disadvantage of the communications in the hybrid code is that MPI messages are longer although there are far less of them. The general trend in MPI programming is that a code with a few long MPI messages is generally quicker than one with several short MPI messages. Here this is clearly not the case.

To determine why this is the case we have developed some test codes. These test codes are similar to the array-100 codes. Again we use a fixed communications pattern as in figure 6.6 and the dynamics stage of the algorithm is mimicked by a *nanosleep* function. The difference between this test code and our previous test codes is that the one component to be cloned is an array that has a length determined by

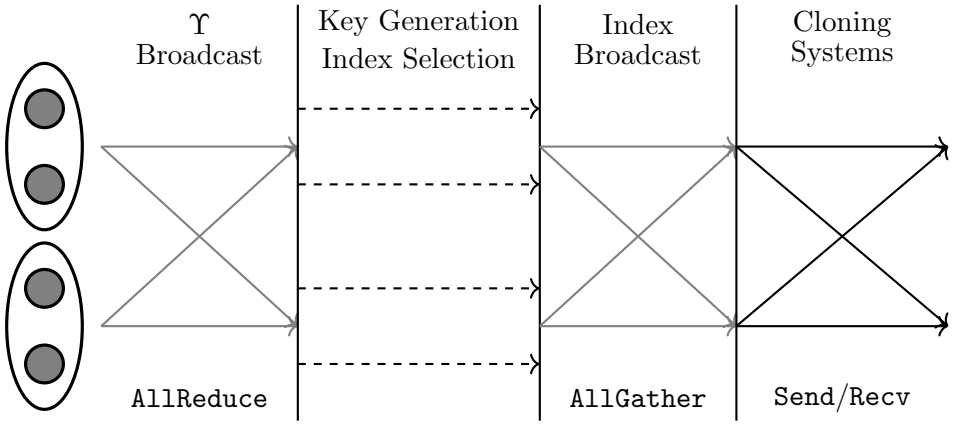


Figure 6.11: Hybrid implementation of the algorithm code run on two nodes (ellipses) and multiple threads (grey circles) with packing of all systems to be sent between each pair of nodes. Arrows are MPI communications, black arrows are pointwise and grey arrows are collective.

an input to the code.

This test code is designed to measure how quickly that the run time of the code increases as the size of the message length increases. We observe in figure 6.13 that as the message size is increased from 10^0 to about 10^2 the run time of the code stays at about a constant. As it is increased beyond this the the run time of the code starts to increase. When there are 10^4 clones per processor as in figure 6.13b increasing messages of size 10^4 by a factor of 5 increases the run time of the code by a factor of 5.26 (3 s.f.).

When we obtain results with $n_c = 10^5$ clones on 64 processors without reduced communications as we have in figure 6.12 we know from figure 6.5 that this corresponds to about 24 clones per message when we pack all systems to be sent between each pair of processors. For systems of size $L = 50$ this corresponds to messages of approximately 1200 integers. When we hybridise the code and have the clones communicated by 4 processors this means that each node packs all the systems to be sent by its 16 processors and all the systems to be received by the 16 processors on the node that it is sending systems to. This means that we have about 16^2 times as many clones per message. This corresponds to messages of approximately 6250 systems and hence approximately 312500 integers for systems of size $L = 50$.

In figure 6.13 when we increase the message size from 5×10^3 to 5×10^4 we also see the run time of the code increase by a factor of 10. Given the rate at which the code run time is increasing with message size in this region of the graph, increasing the message size by another factor of 6 would likely increase the run time of the

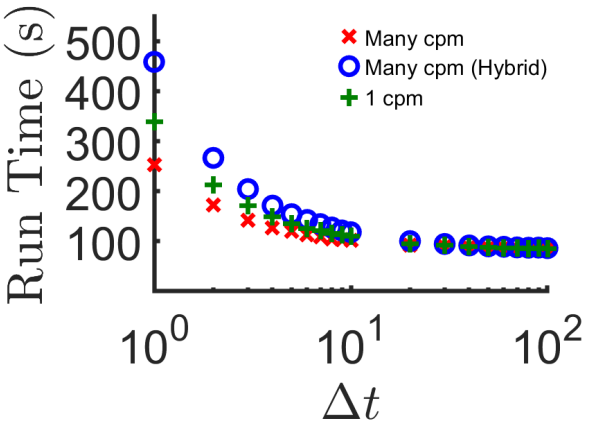


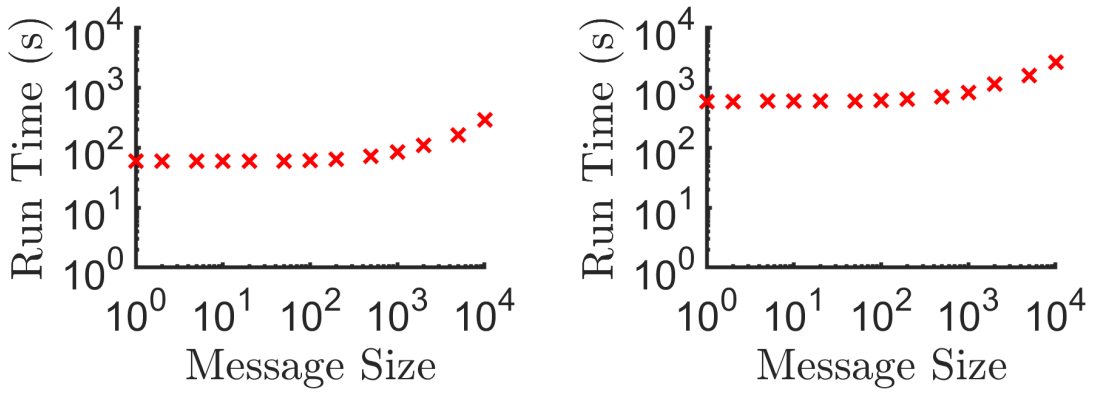
Figure 6.12: Run time of the hybrid code compared with two MPI implementations. Results obtained on 4 nodes of 16 processors per node and averaged over 10 repeats. Results are obtained at the set of parameters in the test case (see table 6.1) at a range of values of the cloning interval and at a bias $sL^2 = 20$.

code by more than a factor of 6. Decreasing the number of messages by a factor of 60 is unlikely to decrease the run time by more than 60 as blocking codes scale linearly with the number of clones and this would only increase the speed of the communications stage.

Overall we conclude that the increase in message size by a factor of 60 for these parameters overrides the decreased number of messages in terms of how it affects the speed of the code. These much larger MPI messages sent by fewer processors explains why the hybrid code in figure 6.11 is slower than the MPI codes with packing. When we increase the number of clones and hence the message size we would expect this observation to be even clearer. Under reduced communications the messages are shorter so we may expect to see a lesser increase in run time from message packing but there are also fewer messages to see an increase in speed by reducing the number of messages.

6.7 Weak Scaling

An important measure of the effectiveness and efficiency of our codes is how well that our codes scale across multiple nodes particularly the code that provides the largest speed-up, the blocking MPI code with reduced communications and which packs multiple clones per message. When we scale the population size n_c with the number of processors that we use this is known as weak scaling [106]. In an ideal implementation the run time would stay the same as the number of processors increases. Here, when we increase the number of processors, the proportion of cloning



(a) $nc_t = 10^3$ (b) $nc_t = 10^4$

Figure 6.13: Run time of the blocking test code described in section 6.6.1 where the component being cloned is an array that may take any size. Results obtained at $t = 10^4$ on 4 nodes of 16 processors per node where nc_t corresponds to the number of clones per thread (processor).

that is performed by MPI communications is increasing as a smaller proportion of systems are expected to be copied internally. Additionally when we have multiple clones per message we have an additional message to be received and sent for each additional processor.

We test the weak scaling of our code at a state point $\lambda = 20$ close to the phase transition in the SSEP. We run the code on systems of 50 sites and $n_c/n_N = 10^5$ so that the algorithm is converged when run on one node. This means that the value of the estimator $\hat{k}$ only depends weakly on the number of systems. We define the abbreviation ppn as meaning processors per node. For both clone selection methods, figure 6.14 shows that the run time increases slowly with the number of nodes under weak scaling. Across the range of n_N from 1 to 8 (16 to 128 processors) the fact that the run time does not increase greatly shows that the code is performing well.

We find these observations to be the case for all three implementations presented in figure 6.14. The ordering of implementations in terms of run time as the number of nodes is increased stays the same and we find that there is only approximately a 5% increase in the run time of the quickest implementation as we increase from 1 node to 8 nodes. This suggests that scaling the code across multiple nodes does not greatly diminish its efficiency. We do not find a point at which the run time of the code becomes rapidly slower or any obvious optimal number of nodes to run the code on.

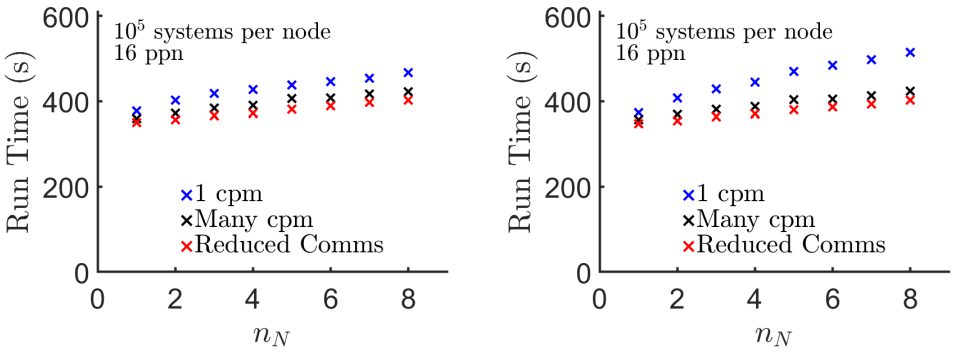


Figure 6.14: Weak Scaling of the run time of the code run for both clone selection methods [iid] (left) and [eq] (right). Results are obtained at the set of parameters in the test case (see table 6.1) at $n_c/n_N = 10^5$ clones per node and at a bias $sL^2 = 20$.

6.8 Derived Data Types

An alternative way to send a system in one MPI message is to use derived data types. Derived MPI data types are user defined data types. To send three vectors and a double in one message we have defined an MPI datatype called *MPI_Lattice* (see Appendix E). One feature of derived datatypes is that the relative memory offset of each component must be known, this varies between instances of the class that defines the system. Therefore each class requires its own datatype. A drawback of this is that multiple systems cannot be sent in one message via this derived datatype method.

We find that the implementation where we use a derived data type is a similar speed but slightly slower than the implementation in which components but not systems are packed. This is a slightly unexpected result as the derived data type implementation does not require the use of packing functions to send one system per message. The reason for this may be that for each system to be sent, the components are still packed by MPI into a buffer to be sent and unpacked from a buffer once they are received. Performing communication with derived data types also requires some additional computation for the derived data types to be produced. As this implementation is slower than the implementation in which components are packed it is also slower than the implementation in which multiple systems are packed into each message. We do not, therefore, use derived data types when obtaining results from our algorithm.

6.9 Compilers and Optimisers

There are various compilers and optimisers that can be used to compile and run our codes. The run time of the code is clearly dependent on the optimiser and compiler that are used. The two OpenMP compilers that we have used here are *g++* and *icpc* and the two MPI compilers are *mpicxx* and *mpiicpc*. We have found in both cases when we run the reduced communication implementation on the SSEP that the intel compilers *icpc* and *mpiicpc* are quicker than *g++* and *mpicxx* for OpenMP and MPI, respectively.

We have also tested a few different optimisers and how the MPI implementation performs when using them. The optimisers we try are $-O1$, $-O2$, $-O3$ and no optimiser and we measure how they scale across multiple nodes. We find that the run times of our implementations when using optimisers $-O2$ and $-O3$ are similar when using up to 5 nodes. When using more than 5 nodes we find that the code has a shorter run time when using the $-O3$ optimiser than when using the $-O2$ optimiser. When we use the $-O1$ optimiser the code is slower than than when we use the $-O2$ and $-O3$ optimisers on any number of nodes. When we use no optimiser we find that when run on any number of nodes that this is slower than the three cases where we use an optimiser.

When obtaining results with the OpenMP method we therefore use the *icpc* compiler. When obtaining results with any of the MPI implementations we are generally obtaining results on 4 nodes of 16 processors. We have found that for this number of nodes that the $-O2$ and $-O3$ optimisers are a similar speed. For simplicity we use the $-O3$ optimiser as this is the better optimiser to use when we are obtaining results on more than 5 nodes. In addition we use the quicker *mpiicpc* compiler when obtaining results with our MPI implementations.

6.10 Computation Summary

In conclusion we have found that the OpenMP implementation is the most efficient. In addition, we have found that the MPI implementation provides a larger speed-up. This may be of use to researchers when they obtain results for large systems or at state points which are difficult to converge. We have found that the MPI implementations can be increased in speed by performing pointwise communication reduction and packing multiple systems into MPI messages.

We have investigated MPI techniques including non-blocking communications and hybridisation. We have found in each case that they do not decrease the run time of the MPI implementations. We have also measured how our MPI implementations

scale across multiple nodes. We have found that when we perform weak scaling the efficiency of the code does not drop off greatly. Also the ordering of the speed of the MPI implementations does not change as we scale from one node to eight nodes under weak scaling.

Chapter 7

Conclusions and Future Work

7.1 Conclusions

We have investigated the probabilities of rare events in physical systems. This has been done using large deviation theory particularly following on from the cloning process first described by Grassberger [51] et al and built upon by Giardinà, Kurchan and Peliti [50] by including modification of dynamics. We have first looked at large deviations of activity in the SSEP, a process particularly studied by Appert-Rolland, Derrida, Lecomte and van Wijland [3] and by Lecomte, Garrahan and others [85]. They have found that when using the algorithm to bias the activity in the SSEP that a phase transition occurs at a particular value of the bias. Our work here also follows on closely from our work on the dynamical phase transition in the SSEP in [19].

The algorithm evolves a large number of systems and we are able to directly measure the variance in activity across these systems when simulating the SSEP as well as directly measuring the activity itself and the mathematical quantity of interest, the large deviation function. For systems of size L, in the limit $L \to \infty$ Appert-Rolland et al [3] and Lecomte, Garrahan et al [85] respectively have obtained theoretical values for these quantities below and above the phase transition. Below the phase transition, Appert-Rolland et al have also found theoretical values for these quantities in finite systems at large values of L and small values of the bias. We have then measured how variables such as the variance in activity scale towards the theoretical $L \to \infty$ values in systems of finite size.

We have found that the algorithmic results are close to the theoretical values that correspond to the $L \to \infty$ limit of the activity and large deviation function for systems of size $L = 20$ and align well for systems of size $L = 50, 80$. The results for the variance in activity show a divergence in the variance in activity that is

consistent with the existence of a phase transition. Our algorithmic values show that the value of the bias at which the peak in the variance in susceptibility occurs decreases in value towards the $L \to \infty$ value predicted by theory as the system size increases. The height of the peak increases towards the value predicted by the theory in the $L \to \infty$ limit as L increases.

We have also investigated how the properties of the system change as the size of the bias is increased from 0 to above the phase transition. At the phase transition we see stable clusters beginning to form. We have measured this by attaining trajectories of the system that show the positions of each particle on the system as well as by measuring the first Fourier component of density. We see that as the bias is increased further the cluster becomes increasingly well-defined and the number of particles in the sparse region outside of the cluster decreases. Physically, this means that in the SSEP when there are rare events of low activity, the system exhibits clusters and high values of the first Fourier component of density.

To obtain results regarding the SSEP we have investigated which algorithmic parameter values to use. One of these parameters is the number of units of time spent implementing the dynamics between each stage that we perform the cloning process, this is referred to as the cloning interval. We have found that when simulating the SSEP that a cloning interval of 10 units of time has the smallest statistical errors associated with it. We have also found that increasing the size of the cloning interval to more units of time than this does not significantly reduce the run time of the code that implements the algorithm despite less cloning processes taking place. When we decrease the size of the cloning interval below this, the increased number of cloning processes increases the run time of this code. As we vary the size of the cloning interval the size of the systematic errors stay the same.

We have also investigated two methods for selecting which systems are going to be copied during the cloning process. One of these is the [iid] method in which markers on a number line of cloning weights are identically independently distributed. The other is the [eq] method in which these markers are equally spaced. We find that these two cloning systems produce systematic errors of the same size. The statistical errors are smaller under the [eq] method for all sizes of cloning interval. In addition the run time of the OpenMP implementation of our code is smaller under the [eq] clone selection method than the [iid] clone selection method. Therefore when obtaining results for the algorithm under the SSEP we use a cloning interval of 10 units of time and the [eq] clone selection method.

The results that we have obtained on the size of the statistical errors varying with the size of the cloning interval are generally going to depend on the system

and process of interest. However, we expect the fact that the [eq] method produces smaller statistical errors than the [iid] method to generally be true for any system and any process. We have found that the systematic errors that we have obtained do not vary with the size of the cloning interval or the choice of clone selection method. We would expect this to also be the case if researchers were to use these clone selection methods to collect results for other systems and processes.

We have also investigated how many units of time are required to obtain accurate results. For each system size $L = 20, 50, 80$ we have investigated how many units of time are required for convergence with respect to a 2% tolerance of an infinite t_{obs} value of the activity obtained by fitting a curve through our algorithmic results. We have investigated the reasons for why the values of the number of units of time required for convergence take the values that they do. We have done this by measuring how long it takes for observables such as the first Fourier component of density $|\delta\rho_1|^2$ and the escape rate r to decay into a time translation invariant regime by measuring their average value over time. We have also measured the timescales associated with the process by measuring the evolutions of the autocorrelations of these observables with respect to the final time t_{obs}.

To obtain accurate results from the algorithm we also require a sufficient number of systems n_{c}. For each system size $L = 20, 50, 80$, we have again fitted a curve through our algorithmic data to obtain an infinite n_{c} value of the activity and we have investigated how many clones are required for convergence with respect to a 2% tolerance of this value. To investigate why we obtain the values that we do for the number of clones required for convergence we have analysed two ways of measuring the distribution of observables. One of these distributions measures the distribution across the population of clones. The other measures the distribution across the population when each system is weighted by the number of descendents that it has at the final time.

From the two methods for measuring the distributions of observables we have derived a lower bound on the number of clones required for convergence. This bound is to assure that at least the peak of a particular one of the distributions is sampled by the other distribution. This bound assumes that both of these distributions have approximately Gaussian distributions. We have used this lower bound to obtain values for how many systems are required for convergence for systems of size $L = 50, 80$ in the SSEP.

The values for the number of clones required for convergence that we have obtained using our convergence criteria based on curve fitting through our algorithmic data are larger by a factor of about 10 than the values that we have obtained using

the lower bound that we have derived. This lower bound is therefore consistent with the other results that we have obtained for convergence with respect to the number of clones. The lower bound that we have derived is a result that could be used by other researchers to obtain a lower bound for the number of clones required for convergence when biasing observables in other processes including in other types of system.

The results that we have obtained for convergence in the SSEP are for systems of size $L = 20, 50, 80$. We have shown that for these systems sizes, the scaling behaviour obtained by Nemoto, Hidalgo and Lecomte is valid. We have only shown, however, that they are valid for the SSEP for systems up to this size. Hidalgo has determined that in the large-L limit, the $1/t_{\text{obs}}$ and $1/n_{\text{c}}$ scalings break down. Therefore at some system size greater than $L = 80$ we expect these scalings to not be valid for the SSEP.

One feature of our results for the SSEP is that a large number of clones is required to obtain results, up to about a million. This is due to the fact that we are considering quite large systems. Some methods [94, 96] may allow results to be obtained with fewer clones and hence results for larger systems than we have considered or for more complicated systems than we have considered. For researchers to efficiently analyse more complicated systems without these methods, they may need to consider smaller system sizes.

As well as large deviations in the SSEP we have also measured large deviations in the Fredkin Process. Firstly, we have looked at large deviations in the activity. We have shown that our algorithm produces accurate results for Fredkin Processes by comparison with theoretical results. Again, we have measured what happens to the clustering of particles as we scale the bias. We have found in typical trajectories of the system and in density profiles of the system that at a negative bias the particles tend to spread out as much as possible. Sometimes Fredkin Processes are represented by a Dyck Path that varies in height as the occupancy of lattice sites vary. When the particles are spread out at a negative bias the area beneath this Dyck Path is small and this corresponds to the centre of mass being near the centre of the lattice.

As the bias is increased to a positive bias we see in the typical trajectories and density profiles that the particles tend to cluster specifically on the left hand end of the lattice. The sites on the left hand end of the lattice are those with indices close to 0. This clustering of particles at the left end of the lattice at a positive bias corresponds to a large area beneath the Dyck Path and a low centre of mass near the left hand end of the lattice. Physically, this means that in Fredkin Processes,

rare events of high activity correspond to particles being spread out and rare events of low activity correspond to particles being clustered on the left hand end of the lattice.

Our results show that when under no bias, the systems under SSEP are different to the systems under the Fredkin Process because under the Fredkin Process particles tend to be positioned towards the left hand side of the lattice in sites with an index close to 0. This is due to the fact that under the Fredkin Process, there must always be more occupied sites than unoccupied sites when reading from left to right across the lattice. This is the also the main difference between the two processes when we scale to a positive bias of the activity. Under the SSEP, particles tend to form into a single cluster. This is also the case under the Fredkin Process, however the cluster is always formed on the left hand boundary of the lattice where the sites have small indices whereas this is not the case under the SSEP.

As with the SSEP we have investigated the large deviation function, activity and susceptibility as a function of a scaling of the bias sL^2. We have found that the values of activity and the large deviation function are similar at this scaling for systems of size $L = 50, 80$ and that the corresponding values for systems of size $L = 20$ are also somewhat similar. We have found that the position of the peak in susceptibility as $L \to \infty$ is at a value of the bias sL^2 that is much smaller than the position of the peak in the SSEP. The value of the bias sL^2 at which the peak in susceptibility exists decreases as the system size increases and may decrease to $sL^2 = 0$.

Although, the activity values and large deviation functions that we measure for the Fredkin Process are similar for different system sizes they do not align as closely as is the case for the SSEP. Also, the values of the scaled bias at which the peaks in susceptibility exist for different system sizes align more closely in the SSEP than they do in the Fredkin Process. When we impose restrictions on the activity in the SSEP as we do in the Fredkin Process, alternative scalings of the bias at which the maximum susceptibilty occurs are expected. This suggests that the sL^2 scaling of the bias against which we plot our results for the SSEP may not be the correct scaling of the bias to plot our results for the Fredkin Process against.

We have also investigated large deviations in the area beneath the Dyck Path. We have found that the properties of the trajectories when we bias this area transition in the opposite way to the properties of the trajectories when we bias the the activity. When we have a negative bias the particles cluster to the left hand boundary of the lattice on sites with indices close to 0. Physically, this is is what corresponds to large values in the area beneath the Dyck Path and a lower value in

the centre of mass. Conversely, at a positive bias the particles tend to spread out and this is physically what happens when there is a low area beneath the Dyck Path corresponding to a higher centre of mass near the centre of the lattice.

As with the SSEP we have then investigated which parameters to use when obtaining results from the Fredkin Process. As with the SSEP we use the [eq] clone selection method to obtain results and a cloning interval of 10 units of time. We again investigate how many units of time are required for the algorithm to converge to within a 2% tolerance of the infinite t_{obs} value of the activity. As when we've obtained results for the SSEP, we obtain this infinite t_{obs} value of the activity by fitting a curve through our data points. By also investigating the average values of observables over time and the evolution over time of autocorrelations of observables with respect to the final time we find that when biasing activity the Fredkin Process has longer time scales associated with it than the SSEP does.

We then go on to consider the clone convergence of the algorithm when biasing activity under the Fredkin Process. We do this by investigating how many clones are required for the results to convergence to within a 2% tolerance of the infinite n_{c} value. We also investigate the different distributions of the activity under the Fredkin Process to understand why the number of clones that are required by the algorithm for convergence takes the value that it does. We find that when biasing the activity under the Fredkin Process we require many fewer systems to converge to an accurate value than when biasing the activity under the SSEP.

As well as investigating the performance of the algorithm when biasing the activity under the Fredkin Process we have also investigated the performance of the algorithm when biasing the area beneath the Dyck Path. Again we fit curves through the data and measure how many units of time are required for the algorithm to converge to within a 2% tolerance of the infinite t_{obs} value. We find that a similar number of units of time are required to converge the algorithm when biasing the area under the Dyck Path to when biasing the activity.

We also fit a curve through the data to measure how many systems are required for the algorithm to converge to within a 2% tolerance of the infinite n_{c} value. We find that a similar number of clones are required for convergence when biasing the area beneath the Dyck Path to when biasing the activity. The reason that a similar number of units of time are required for both observables may be because in both cases the algorithm biases to trajectories with configurations in which particles are clustered to the left hand boundary and which exhibit the longest timescales. The number of clones required for convergence may be similar for the two observables because the p_{ave} and p_{end} distributions have a similar amount of overlap for the

values of the bias that we have considered.

The results that we have obtained for convergence in the Fredkin Process are for systems of size $L = 20, 50, 80$ when biasing the activity and $L = 20, 50$ when biasing the area beneath the Dyck Path. In both cases, we have found that for the systems sizes that we have considered, the scaling behaviour obtained by Nemoto, Hidalgo and Lecomte is valid. We have not shown, however, that they are valid for Fredkin Processes for systems larger than the ones that we have considered. Hidalgo has determined that in the large-L limit, the $1/t_{\text{obs}}$ and $1/n_{\text{c}}$ scalings break down. Therefore at some system size greater than $L = 80$ when biasing the activity and $L = 50$ when biasing the area beneath the Dyck Path, we expect these scalings to not be valid for the Fredkin Process.

We have also investigated various parallelisation techniques for implementing the algorithm. The first of these that we have tried is an OpenMP implementation. We have found that this is the most efficient of the parallelisation techniques that we use and is able to obtain efficiencies of 98.8%. We are only able to run this implementation on one node of 16 processors at most. In this set-up we obtain a speed-up of 15.8. We have also found that a simple MPI implementation of the algorithm produces larger speed-ups as it can be run on multiple nodes. A larger speed-up of 48.6 is obtained when running this implementation on 4 nodes of 16 processors per node. This speed-up corresponds to an efficiency of 75.9%, less than the OpenMP implementation.

One way to increase the speed of the MPI implementations further is to use packing functions. This means that a system made of several components can be packed into one message. It also means that when multiple systems are sent from one processor to another that they can all be packed into one MPI message. When packing a system's components the speed-up of the code increases to 49.8 corresponding to an efficiency of 77.8%.

Another method to increase the speed of the MPI implementation is to reduce the number of MPI communications by pointwise cancellation of the number of systems sent between each pair of processors. This means that many systems that would have been sent via MPI between each pair of processors are instead copied internally. We find that the number of systems sent by each processor and between each pair of processors decreases significantly. The number of messages sent decreases by a factor of 2 and the average message size decreases from containing around 25 systems to containing around 5 systems.

We have also investigated the effect of using non-blocking communications when implementing our code. These are codes in which computation such as the

dynamics can be executed whilst MPI communications such as the communications stage occur simultaneously. We have constructed test codes to measure how the run time of the code varies as the number of systems per processor is varied. We find that none of the various non-blocking implementations that we have produced are as fast as the blocking implementation when we have the number of clones per processor that would use when obtaining results from the algorithm.

Another approach that we have used to increase the speed of our code is to use OpenMP-MPI hybridisation. This is where within each node the program is parallelised with OpenMP and then MPI is used to communicate systems between nodes. When we do this we find that the code is not as quick as the simpler MPI implementations that employ the packing of systems and of components.

We measure how well MPI implementations scale across multiple nodes. We find that as we increase the number of nodes that we run the code on from one to eight with a fixed number of systems per node there is only approximately a 5% increase in the run time of the quickest code. Another technique we use to attempt to increase the speed of our code is the use of Derived Data Types. This is where we define a data type that contains all of the components associated with a system so that the whole system can be sent in one MPI message without packing. We do not use derived data types as we have found that using them is marginally slower than our previous implementation that uses component packing. We have also tested various compilers and optimisers to determine which are the best with which to run our code. We have concluded that the *icpc* compiler is the best with which to run our OpenMP code in terms of lowest run time. We have also found that when running MPI implementations the optimiser which produces the largest speeds is $-O3$ and the most efficient MPI compiler to use is *mpiicpc*.

The results that we have obtained for the speed and efficiency of different computational techniques are for a specific test case where we implement the SSEP. We have found that the OpenMP code is the most efficient code and we anticipate that this should be the case regardless of the amount of time spent simulating dynamics compared to the amount of time performing the cloning process. This is because the OpenMP overheads induced by the cloning process are always smaller than the corresponding MPI overheads. Therefore for any system and any process the OpenMP implementation should always be the most efficient.

When investigating the speed-up obtained by MPI implementations we have found that packing components and systems has always increased the speed of the code. This should be the case for any system and for any process as it reduces the number of MPI communications. Similarly the pointwise cancellation procedure

should also increase the speed of the code for any system and for any process as it reduces the quantity and size of MPI communications. For some systems and processes, our results suggest that non-blocking implementations may be faster than blocking implementations when there are sufficiently few clones required for convergence. Our results also suggest that for sufficiently small systems, the size of MPI messages may be small enough that an MPI-OpenMP hybrid code would be quicker than other MPI codes, however, communications being performed by a small number of processors may still lead to large run times.

7.2 Future Work

There are several ways in which the results obtained in this thesis could be built upon. One of these is to explore how the algorithm can be used to obtain results for other processes. One type of process of interest may be those considered by Evans, Kafri, Levine and Mukamel [43] involving different types of interacting particles such as positive and negative particles. Other processes of interest may be those in which particles may proliferate and/or the system may grow with time. Processes like this are relevant in cell colonization during embryonic development [11]. Various birth defects can result from unsuccessful colonization and so it is of importance to be able to use models to calculate probabilities associated with this.

Another type of process that it may be interesting to test the algorithm on is higher-dimensional processes than the ones that we have tested here such as particles moving on grids or networks. Large deviations have previously been studied in the rare events of complex networks [33, 81] and it may be insightful to compare results from our implementation of the algorithm to results obtained by their methods. For each of these processes it would be of interest to compare them with the processes that we have already considered here, the SSEP and the Fredkin Process, in terms of how the accuracy and run time of the algorithm are dependent on the size of the cloning interval Δt and how the observable time t_{obs} and population size n_{c} convergences vary with the definitions of the process.

We have so far estimated averages of observables by using summation (4.3). As discussed in section 4.3, this summation is dominated by contributions from the TTI regime, introduced in section 2.5. There are however contributions from the transient regimes which incur errors in our estimations of these averages. As discussed in section 4.3, alternatively one could estimate $\mathcal{K}$ from the plateau values of $\left\langle K^{\beta} \right\rangle / L$. This may be a useful strategy for obtaining averages from the algorithm in the future.

One attribute of this region of the large deviations of the SSEP that is not understood is the rate at which the peak of the susceptibility $\mathcal{X}(\lambda)$ increases towards the large-L limit as the number of sites L is increased for finite L. This would be an interesting way to add to the results that we have obtained. Furthermore, the value of λ at which the values of $\mathcal{X}(\lambda)$ crossover for finite values of L is not understood and would be another interesting area of research in the future. The correct scaling of the bias for our results for large deviations in activity in the Fredkin Process to be plotted against is also unknown and would be a further informative area of future research.

As well as other physical processes there are other computational techniques that it would be interesting to use the code that implements the algorithm to test. Currently we are using two-sided MPI communication where both processors post a send or receive for each MPI communication. An alternative is to use one-sided communication which may under MPI be implemented in several different ways [55]. Under one sided-communication each processor places messages to be sent in a remote memory access window which other processors are then able to pick out. Another computational technique that could be utilised is to advance the communications reduction further. This can be done by performing the same cancellation procedure as we have used between processors to reduce the number of systems being sent between nodes. We may find it quicker for systems to be communicated intranodally than between nodes.

Overall, the cloning algorithm that we have investigated the performance of and have used to study the SSEP and the Fredkin Process, can in principle be applied to a wide range of processes across many fields such as geology, finance, chemistry, biology and cosmology. In each case the cloning algorithm can be used to calculate the probabilities of rare events and the values of associated quantities such as the large deviation function and to characterise rare events to determine the conditions in which they occur. If by scaling our MPI implementations of the algorithm across a larger numbers of processors we are able to obtain results more quickly this would give us the opportunity to study bigger and more complicated systems. This would be a productive and informative way to build on the results that we have obtained.

Appendix A

Relating $\psi(s)$ to $\pi(a)$ Using the Observable Value at time t

The purpose of this appendix is to derive the relation between the large deviation function $\psi(s)$ and the rate function $\pi(a)$ by considering the change in the value of the observable between configurations. In addition by considering the modified rates of changes between configurations we state a master equation to describe the true dynamics and modified dynamics and derive a master equation to describe the evolution of the path measure.

We denote $P(A,t)$ as the probability of a system having observable value A at time t. This is related to the probability $P(C,A,t)$ by

$$P(A,t) = \sum_{C} P(C,A,t), \tag{A.1}$$

where $P(C,A,t)$ is the probability of being in configuration C with observable value A at time t. The probability of having observable value A can be written as the proportion of all trajectories that give this observable value. By defining A_t as the value of the observable at time t, $P(C,A,t)$ is re-stated as

$$P(A,t) = \left\langle \delta\big(A - A_t\big) \right\rangle, \tag{A.2}$$

by taking the average over all trajectories of delta functions that pick out trajectories that lead to the observable having a value of A. This delta function can be stated using its integral representation

$$P(A,t) = \frac{1}{2\pi i} \int_{-i\infty}^{+i\infty} ds \ \left\langle \exp(-s[A_t - ta]) \right\rangle, \tag{A.3}$$

using the definition $a = A/t$. From the definitions of the large deviation function (2.4) and the partition function (2.3) we state

$$\exp(t\psi(s)) = \langle \exp(-sA_t) \rangle \,, \tag{A.4}$$

in the limit $t \to \infty$ which is substituted into equation (A.3)

$$P(A,t) = \frac{1}{2\pi i} \int_{-i\infty}^{+i\infty} ds \;\; \exp(t[\psi(s) + sa]), \tag{A.5}$$

to obtain a link between $\psi(s)$ and the probability of having a particular observable value. We compute the integral to obtain the solution

$$P(A,t) \sim \exp\left(-t \max_s [-\psi(s) - sa]\right), \tag{A.6}$$

in the limit of $t \to \infty$ by assuming that $\psi(s)$ is finite in this limit and using the method of steepest descent. This assumes that the imaginary contour line can be deformed to the real line. This is the approach of Giardinà, Kurchan and Peliti [50] to compute the integral in equation (A.5). From our definition (2.1) of $\pi(a)$ and equation (A.6) we obtain

$$\pi(a) = \max_s [-\psi(s) - sa], \tag{A.7}$$

by assuming that $\pi(a)$ is convex [48] which one may assume in the absence of dynamical phase transitions [19]. We also know that the rate functions $\pi(a)$ that we consider are convex as we are considering discrete Markov chains. This means that $\pi(a)$ is the Legendre transform of $\psi(s)$.

From here we consider type $\mathcal{A}$ observables that change in value every time that the system changes configuration as we have defined them in definition (2.7). We follow the approach of Lecomte and Tailleur [86]. The probability $P(C, A, t)$ is defined as the probability of being in configuration C with observable value A at time t. These probabilities evolve according to a master equation

$$\partial_t P(C,A,t) = \sum_{C'} W(C' \to C) P(C', A - \alpha(C', C), t) - r(C) P(C, A, t), \tag{A.8}$$

where $\alpha(C, C')$ is the change in the value of the observable between configuration C and configuration C', $W(C' \to C)$ are the transition rates and $r(C)$ is the escape rate. The modified probabilities $\hat{P}(C, A, t)$ evolve according to a similar master

equation but with modified transition rates $W_s(C' \to C)$ and a modified escape rate $r_s(C)$. The escape rate is defined by

$$r(C) = \sum_{C'} W(C \to C'). \tag{A.9}$$

A statistical weight is defined by

$$\tilde{P}(C, s, t) = \sum_{A} \exp(-sA) P(C, A, t), \tag{A.10}$$

and we want to find a master equation that describes how this statistical weight changes over time. A sensible approach is to take the Laplace transform of both sides of master equation (A.8)

$$\begin{aligned}\sum_{A} \partial_t \left[\exp(-sA) P(C, A, t)\right] = &\sum_{C'} \sum_{A} \exp(-sA) W(C' \to C) P(C', A - \alpha(C', C), t) \\ &- \sum_{A} \exp(-sA) r(C) P(C, A, t),\end{aligned} \tag{A.11}$$

by multiplying all terms by e^{-sA} and summing over all values of A. We then introduce the modified transition rates

$$W_s(C \to C') = \exp(-s\alpha(C, C')) W(C \to C'), \tag{A.12}$$

which we substitute into the transformed master equation

$$\begin{aligned}\partial_t \tilde{P}(C, s, t) = &\sum_{C'} \sum_{A} \exp(-s(A - \alpha(C', C))) W_s(C' \to C) P(C', A - \alpha(C', C), t) \\ &- r(C) \tilde{P}(C, s, t),\end{aligned} \tag{A.13}$$

where we have substituted in $\tilde{P}(C, s, t)$. The introduction of the modified transitions $W_s(C' \to C)$ then allows us to perform the sum over all possible values of A in the first term of our transformed equation and obtain our final version

$$\partial_t \tilde{P}(C, s, t) = \sum_{C'} W_s(C' \to C) \tilde{P}(C', s, t) - r(C) \tilde{P}(C, s, t). \tag{A.14}$$

This master equation has no steady state solution but it does have solutions that behave for long times as $\tilde{P}(C, s, t) = Q(C, s) \exp(\psi(s) t)$. Substituting this into master equation A.14 gives our usual eigenvalue problem with ψ as the eigenvalue and Q as the eigenvector.

Appendix B

Calculation of Hop Success Rates

In this appendix we determine that hops in the SSEP are successful at the correct rate. We do this by considering probability distributions of the same form as those from which time steps are drawn.

At each time step we pick a random one of N particles, each having a probability of being picked of $\frac{1}{N}$. We also pick a direction, the probability of picking left is $\frac{p}{p+q}$ and of picking right is $\frac{q}{p+q}$, where p and q are our left rate and right rate. If the particle can move to the neighbouring site in the direction that we have picked (i.e. if the neighbouring site is empty) then the particle is moved. Regardless of whether the move takes place

$$P(\Delta t) = N(p+q)\exp(-N(p+q)\Delta t), \tag{B.1}$$

describes the probability distribution of Δt, which is the amount by which the time is incremented. If we are in configuration C at time t_0, we want to determine the probability, $P(t_1|C, t_0)$, that a hop occurs at time t_1. To do this, we define R as the number of rejected moves between t_0 and t_1. These rejections occur at times: s_0, s_1 ... s_R.

We define the freedom $f_i(C)$ of each particle in configuration C as the number of empty sites that are neighbouring the site that it occupies. The total freedom is therefore equal to the total number of possible configuration changes. For every cluster of particles there is one particle hop to the left that is possible and one to the right. So half of the configuration changes are hops to the left and half are hops to the right. The total of all rates is therefore $[p/2 + q/2]$ multiplied by the total number of possible configuration changes. We have that

$$r(C) = \sum_{C'} W(C \to C') = \frac{(p+q)}{2} \sum_i f_i(C), \tag{B.2}$$

which relates the escape rate $r(C)$ of configuration C to the particle freedoms. Clearly the probability of picking a successful hop is $\frac{\sum_i f_i(C)}{2N}$. Using equation (B.2) we obtain that

$$\frac{(p+q)N - r(C)}{(p+q)N}, \tag{B.3}$$

is the probability of picking a move that is rejected. We use equation (B.1) to calculate the probability of attempts occurring at each value of s and we obtain the probability that a hop occurs at time t_1

$$P(t_1|C,t_0) = \sum_{R=0}^{\infty} \int_{t_0}^{t_1} ds_1 \int_{s_1}^{t_1} ds_2 \int_{s_{R-1}}^{t_1} ds_R \; P_f(s_1|t_0) \; P_f(s_2|s_1) \; ... \\ P_f(s_R|s_{R-1}) \; N \; (p+q) \; \exp(-N(p+q)(t_1 - s_R)) \frac{r(C)}{(p+q)N}, \tag{B.4}$$

by summing over paths with all possible R and s variables. The final term in equation (B.4) is the probability of a successful hop occurring after our final rejected move at time s_R. The $P_f(b|a)$ terms in equation (B.4) are the probabilities of a failed attempt occurring at time b given that a move attempt occurred at time a. These are defined by

$$P_f(b|a) = \frac{(p+q)N - r(C)}{(p+q)N} (p+q)N \exp(-N(p+q)(b-a)) \tag{B.5}$$

$$= [(p+q)N - r(C)] \exp(-N(p+q)(b-a)), \tag{B.6}$$

which are obtained by multiplying the probability of a hop occurring with the probability of it being rejected. We can then write

$$P(t_1|C,t_0) = \exp(-(p+q)N(t_1 - t_0)) \\ r(C) \; \sum_{R=0}^{\infty} [(p+q)N - r(C)]^R \int_{t_0}^{t_1} ds_1 \int_{s_1}^{t_1} ds_2 \int_{s_{R-1}}^{t_1} ds_R, \tag{B.7}$$

by using the fact that the exponential terms can be collected into a term that does not depend on s, and re-writing the product of the other terms. We can try to do the integrals (note the ranges) but the integrand (which is equal to 1) is symmetric under interchange of all s arguments and so we can replace all the lower limits by

t_0 as long as we multiply by $\frac{1}{R!}$. Then we obtain equation

$$P(t_1|C,t_0) = \exp(-(p+q)N(t_1-t_0)) \\ r(C)\sum_{R=0}^{\infty}\left([(p+q)N - r(C)]^R (t_1-t_0)^R \frac{1}{R!}\right), \tag{B.8}$$

as each integral yields a factor of $(t_1 - t_0)$. The sum is the series expansion of an exponential which gives us

$$P(t_1|C,t_0) = \exp(-(p+q)N(t_1-t_0))r(C)\exp([(p+q)N - r(C)](t_1-t_0)) \tag{B.9}$$

$$= r(C)\exp(-r(C)(t_1-t_0)), \tag{B.10}$$

as the probability of a successful hop occurring at t_1, given that we change to configuration C at time t_0. This means that hops occur at the correct rate of $r(C)$. Under modified dynamics we used modified rates and the same arguments apply so we obtain

$$\hat{P}(t_1|C,t_0) = r_s(C)\exp(-r_s(C)(t_1-t_0)), \tag{B.11}$$

as the probability of a successful hop occurring at t_1, given that we change to configuration C at time t_0. Therefore, hops occur at the correct rate $r_s(C)$.

Appendix C

Satisfying the Master Equation

In this appendix we show that given the probabilities that we have previously determined of a hop succeeding at a given point in time and the rate at which successful hops occur, master equations (A.8) and (A.14) are satisfied.

The probability that any stochastic process is in a state after an infinitesimal amount of time is given by

$$P(\gamma, t+dt) = P(\gamma, t) + \sum_{\gamma'} \left[P(\gamma', t)\ \text{rate}(\gamma' \to \gamma) dt\right] - \sum_{\gamma'} \left[P(\gamma, t)\ \text{rate}(\gamma \to \gamma') dt\right], \tag{C.1}$$

where γ refers to a state and γ' refers to any other state. In our case the state is comprised of the value of the observable A and the configuration C so

$$P(C, A, t+dt) = P(C, A, t) + \sum_{C'} \left[P(C', A - \alpha(C', C), t)\ \text{rate}(C' \to C) dt\right] - \sum_{C'} \left[P(C, A, t)\ \text{rate}(C \to C') dt\right], \tag{C.2}$$

describes the rate at which we move between configurations and observable values. We have obtained in Appendix B that when the dynamics are implemented, the probability distribution of the time t^* at which we leave configuration C is

$$P(t^*) = r(C) \exp(-r(C)(t^* - t')), \tag{C.3}$$

if we enter this configuration at time t'. Given the memory-less property of exponential distributions, the probability at any time t of exiting configuration C at time t^* is

$$P(t^* | \text{no hop } t' \to t) = r(C) \exp(-r(C)(t^* - t)). \tag{C.4}$$

The average value of time between hops of this exponential probability distribution is $\frac{1}{r(C)}$. This means that the rate of hops is $r(C)$. When there is a hop from C, the probability of it going to C' is $\frac{W(C\to C')}{r(C)}$. Therefore the transition rate that we implement from C to C' is $W(C \to C')$. Hence, equation(C.2) becomes

$$P(C, A, t + dt) = P(C, A, t) + \sum_{C'} [P(C', A - \alpha(C', C), t)\ W(C' \to C)dt] \quad \text{(C.5)}$$
$$- P(C, A, t)\ r(C)dt,$$

noting that $r(C) = \sum_{C'} W(C \to C')$. By using the first forward difference operator

$$\partial_t P(C, A, t) = \frac{P(C, A, t + dt) - P(C, A, t)}{dt}, \quad \text{(C.6)}$$

where dt is an infinitesimal amount of time we re-arrange equation (C.5) to obtain the master equation

$$\partial_t P(C, A, t) = \sum_{C'} W(C' \to C)P(C', A - \alpha(C', C), t)\ - r(C)P(C, A, t), \quad \text{(C.7)}$$

which is the same as master equation (A.8) and hence the master equation is satisfied by attempting hops at the rates at which we do. The same arguments apply for the equivalent master equation for the modified rates.

Appendix D

Path Measures

The purpose of this appendix is to derive the cloning weights (factors) that we use when simulating the algorithm. We derive the weights required for when the SSEP is modified and for when processes are not modified. During the modified dynamics phase each system is evolved from the start time of each cloning interval, t_0. We normalise

$$\hat{P}(C, s, t_0) = \frac{\tilde{P}(C, s, t_0)}{Z(s, t_0)}, \tag{D.1}$$

at the start of each cloning interval to be proportional to the path measure and these probabilities remain normalised throughout the implementation of the modified dynamics. Under modified dynamics it is shown in Appendix B that

$$\hat{P}(t_1|C, t_0) = r_s(C) \exp(-r_s(C)(t_1 - t_0)), \tag{D.2}$$

is the probability of a hop occurring at time t_1 if we're in configuration C at time t_0. When we change configuration, the probability of changing to configuration C_1 is given by $\frac{W_s(C \to C_1)}{r_s(C)}$. By taking into account the probability of being in configuration C_0 at time t_0 we can hence determine the probability

$$\hat{P}(C_0 \to C_1, s, t_0 \to t_1) = \frac{\tilde{P}(C_0, s, t_0)}{Z(s, t_0)} W_s(C_0 \to C_1) \exp(-r_s(C_0)(t_1 - t_0)), \tag{D.3}$$

of following a path from C_0 at t_0 to C_1 at t_1. The cloning interval is defined as being of length Δt, in which K hops occur with the final hop occurring at time t_K. Using the same logic as before, consider the case where the system evolves continually

along a path, where the path probabilities are

$$\hat{P}(C_0 \to \ldots \to C_K, s, t_0 \to \ldots \to t_K) = \frac{\tilde{P}(C_0, s, t_0)}{Z(s, t_0)} \left[\prod_{i=0}^{K-1} W_s(C_i \to C_{i+1})\right] \exp\left(\sum_{i=0}^{K-1} -r_s(C_i)(t_{i+1} - t_i)\right), \tag{D.4}$$

under modified dynamics and the system changes to configuration C_i at time t_i. We need to determine the probability of these configuration changes occurring at particular times and then no more occurring during this cloning interval, i.e. between time t_K and time $t_0 + \Delta t$. If a hop has occurred at time t_K the probability of a hop occurring before time $t_0 + \Delta t$ by integrating equation (D.2) from t_K to $t_0 + \Delta t$ is calculated

$$\hat{P}(\text{hop before } t_0 + \Delta t | C_K, s, t_K) = \int_{t_K}^{t_0+\Delta t} r_s(C_K) \exp(-r_s(C_K)(t - t_K))dt \tag{D.5}$$

$$= 1 - \exp(-r_s(C_K)([t_0 + \Delta t] - t_K)). \tag{D.6}$$

Knowing the probability that there *is* a hop within this time period, there is a probability

$$\hat{P}(\text{no hop before } t_0 + \Delta t | C_K, s, t_K) = \exp(-r_s(C_K)([t_0 + \Delta t] - t_K)), \tag{D.7}$$

of there *not* being a hop between t_K and $t_0 + \Delta t$. It is then possible to state the probability of a system following a particular path from the start of the cloning interval to the end of the cloning interval

$$\hat{P}(C_0 \to \ldots \to C_K \to C_K, s, t_0 \to \ldots \to t_K \to t_0 + \Delta t) = \frac{\tilde{P}(C_0, s, t_0)}{Z(s, t_0)} \left[\prod_{i=0}^{K-1} W_s(C_i \to C_{i+1})\right] \exp\left(-\int_{t_0}^{t_0+\Delta t} r_s(C)\, dt\right), \tag{D.8}$$

obtained by multiplying equation (D.4) by equation (D.7). This then gives us the probability of following a particular path from equation (D.4) and then of not hopping again before the end of the cloning interval from equation (D.7). Note that in doing this we replace the summation of the escape rate multiplied by the time step size with the integral of the escape rate. These quantities are equivalent to one another. When considering the $\tilde{P}(C, s, t)$ measure the partition function $Z(s, t_0)$ is not used to normalise at the beginning of each cloning interval as the total of $\tilde{P}(C, s, t)$ is free to be greater than or less than unity. This would correspond to

a growing or shrinking population if we were not keeping the population constant. When evolving the systems by the true dynamics as opposed to the modified dynamics, $r_s(C)$ is replaced by $r(C)$. The $\tilde{P}(C, s, t)$ measure also evolves by $r(C)$ as opposed to $r_s(C)$. The statistical weight of each path is given by

$$\tilde{P}(C_0 \to \ldots \to C_K \to C_K, s, t_0 \to \ldots \to t_K \to t_0 + \Delta t) = \tilde{P}(C_0, s, t_0) \left[\prod_{i=0}^{K-1} W_s(C_i \to C_{i+1}) \right] \exp\left(- \int_{t_0}^{t_0+\Delta t} r(C)\ dt \right), \quad \text{(D.9)}$$

where $r_s(C)$ is replaced by $r(C)$. By using '*path*' to denote a particular set of configuration changes at particular times $C_0 \to \ldots \to C_K \to C_K$, equation (D.9) is divided by equation (D.8) to obtain

$$\tilde{P}(s, \text{path}) = \Upsilon\ \hat{P}(s, \text{path})\ Z(s, t_0), \quad \text{(D.10)}$$

a relation between the two ways of measuring each path. In equation (D.10) the cloning factors

$$\Upsilon = \exp\left(\int_{t_0}^{t_0+\Delta t} [r_s(C) - r(C)]\ dt \right), \quad \text{(D.11)}$$

are defined. If the systems are evolving under the true dynamics with importance sampling, the probability of following a particular path is

$$P(C_0 \to \ldots \to C_K \to C_K, t_0 \to \ldots \to t_K \to t_0 + \Delta t) = \frac{\tilde{P}(C_0, s, t_0)}{Z(s, t_0)} \left[\prod_{i=0}^{K-1} W(C_i \to C_{i+1}) \right] \exp\left(- \int_{t_0}^{t_0+\Delta t} r(C)\ dt \right). \quad \text{(D.12)}$$

We would then obtain a different cloning factor, Φ, relating these two path measures to one another

$$\tilde{P}(s, \text{path}) = \Phi\ P(\text{path})\ Z(s, t_0), \quad \text{(D.13)}$$

by dividing equation (D.9) by equation (D.12). The cloning factors under the true dynamics are

$$\Phi = \frac{\prod_{i=0}^{K-1} W_s(C_i \to C_{i+1})}{\prod_{i=0}^{K-1} W(C_i \to C_{i+1})} = \prod_{i=0}^{K-1} \exp(-s\alpha(C_i \to C_{i+1})) = \exp(-sA). \quad \text{(D.14)}$$

Appendix E

Derived Data Type for a One-Dimensional Lattice

In section 6.8 we have described a derived MPI datatype that we have defined for a one-dimensional lattice that includes three vectors and a double. The *C++* code for this *MPI_Lattice* datatype is displayed below.

Listing E.1: Derived *MPI_Lattice* datatype

```
int count = 4;
int blocklengths[4] = {ptcls, sites, ptcls, 1};
MPI_Aint ploc, sloc, floc, eloc;
MPI_Datatype types[] = {MPI_INT, MPI_INT, MPI_INT, MPI_DOUBLE};

for (i = 0; i < cpp; i++)
{
  k = i * numproc + myid;
  if (k < nlat)
  {
   MPI_Get_address(lats[i]->p.data(), &ploc);
   MPI_Get_address(lats[i]->s.data(), &sloc);
   MPI_Get_address(lats[i]->freedom.data(), &floc);
   MPI_Get_address(&lats[i]->esc, &eloc);
   MPI_Aint displacements[] =
     {ploc-eloc, sloc-eloc, floc-eloc, 0};
   MPI_Type_create_struct(count, blocklengths,
     displacements, types, &(infos[i]->MPI_Lattice));
   MPI_Type_commit(&(infos[i]->MPI_Lattice));
```

```
    MPI_Get_address(lats[i+cpp]->p.data(), &ploc);
    MPI_Get_address(lats[i+cpp]->s.data(), &sloc);
    MPI_Get_address(lats[i+cpp]->freedom.data(), &floc);
    MPI_Get_address(&lats[i+cpp]->esc, &eloc);
    displacements[0] = ploc-eloc;
    displacements[1] = sloc-eloc;
    displacements[2] = floc-eloc;
    MPI_Type_create_struct(count, blocklengths,
      displacements, types, &(infos[i+cpp]->MPI_Lattice));
    MPI_Type_commit(&(infos[i+cpp]->MPI_Lattice));
  }
}
```

Bibliography

[1] R Allen, P Warren, and Pieter Rein Ten Wolde, *Sampling rare switching events in biochemical networks*, Physical Review Letters **94** (2005), no. 1, 018104.

[2] J Anderson, *A random-walk simulation of the schrödinger equation: H+ 3*, The Journal of Chemical Physics **63** (1975), no. 4, 1499–1503.

[3] C Appert-Rolland, B Derrida, V Lecomte, and F van Wijland, *Universal cumulants of the current in diffusive systems on a ring*, Physical Review E **78** (2008), no. 2, 021122.

[4] Y Baek, Y Kafri, and V Lecomte, *Dynamical symmetry breaking and phase transitions in driven diffusive systems*, Physical Review Letters **118** (2017), no. 3, 030604.

[5] P Balaji, A Chan, W Gropp, R Thakur, and E Lusk, *Non-data-communication overheads in mpi: analysis on blue gene/p*, European Parallel Virtual Machine/Message Passing Interface Users' Group Meeting, Springer, 2008, pp. 13–22.

[6] C Beck and F Schögl, *Thermodynamics of chaotic systems: an introduction*, no. 4, Cambridge University Press, 1995.

[7] N Beisert, A Tseytlin, and K Zarembo, *Matching quantum strings to quantum spins: One-loop vs. finite-size corrections*, Nuclear Physics B **715** (2005), no. 1-2, 190–210.

[8] L Bertini, A De Sole, D Gabrielli, G Jona-Lasinio, and C Landim, *Current fluctuations in stochastic lattice gases*, Physical Review Letters **94** (2005), no. 3, 030601.

[9] ———, *Non equilibrium current fluctuations in stochastic lattice gases*, Journal of Statistical Physics **123** (2006), no. 2, 237–276.

[10] ———, *Macroscopic fluctuation theory*, Reviews of Modern Physics **87** (2015), no. 2, 593.

[11] B Binder, K Landman, D Newgreen, and J Ross, *Incomplete penetrance: The role of stochasticity in developmental cell colonization*, Journal of Theoretical Biology **380** (2015), 309–314.

[12] T Bodineau and B Derrida, *Distribution of current in nonequilibrium diffusive systems and phase transitions*, Physical Review E **72** (2005), no. 6, 066110.

[13] ———, *Cumulants and large deviations of the current through non-equilibrium steady states*, Comptes Rendus Physique **8** (2007), no. 5, 540–555.

[14] T Bodineau, V Lecomte, and C Toninelli, *Finite size scaling of the dynamical free-energy in a kinetically constrained model*, Journal of Statistical Physics **147** (2012), no. 1, 1–17.

[15] T Bodineau and C Toninelli, *Activity phase transition for constrained dynamics*, Communications in Mathematical Physics **311** (2012), no. 2, 357–396.

[16] P Bolhuis, D Chandler, C Dellago, and P Geissler, *Transition path sampling: Throwing ropes over rough mountain passes, in the dark*, Annual Review of Physical Chemistry **53** (2002), no. 1, 291–318.

[17] A Bortz, M Kalos, and J Lebowitz, *A new algorithm for monte carlo simulation of ising spin systems*, Journal of Computational Physics **17** (1975), no. 1, 10–18.

[18] F Bouchet and J Reygner, *Generalisation of the eyring–kramers transition rate formula to irreversible diffusion processes*, Annales Henri Poincaré, vol. 17, Springer, 2016, pp. 3499–3532.

[19] T Brewer, S Clark, R Bradford, and R Jack, *Efficient characterisation of large deviations using population dynamics*, Journal of Statistical Mechanics: Theory and Experiment **2018** (2018), no. 5, 053204.

[20] M Burman, D Carpenter, and R Jack, *Emergence of particle clusters in a one-dimensional model:connection to condensation processes*, Journal of Physics A: Mathematical and Theoretical **50** (2017), no. 13.

[21] N Cancrini, F Martinelli, C Roberto, and C Toninelli, *Kinetically constrained lattice gases*, Communications in Mathematical Physics **297** (2010), no. 2, 299–344.

[22] F Carollo, J Garrahan, I Lesanovsky, and C Pérez-Espigares, *Fluctuating hydrodynamics, current fluctuations, and hyperuniformity in boundary-driven open quantum chains*, Physical Review E **96** (2017), no. 5, 052118.

[23] E Carter, G Ciccotti, J Hynes, and R Kapral, *Constrained reaction coordinate dynamics for the simulation of rare events*, Chemical Physics Letters **156** (1989), no. 5, 472–477.

[24] M Cavallaro and R Harris, *A framework for the direct evaluation of large deviations in non-markovian processes*, Journal of Physics A: Mathematical and Theoretical **49** (2016), no. 47, 47LT02.

[25] B Chapman, G Jost, and R Van Der Pas, *Using openmp: portable shared memory parallel programming*, vol. 10, MIT press, 2008.

[26] X Chen, E Fradkin, and W Witczak-Krempa, *Gapless quantum spin chains: multiple dynamics and conformal wavefunctions*, Journal of Physics A: Mathematical and Theoretical **50** (2017), no. 46, 464002.

[27] R Chetrite and H Touchette, *Nonequilibrium markov processes conditioned on large deviations*, Annales Henri Poincaré, vol. 16, Springer, 2015, pp. 2005–2057.

[28] L Chong, A Saglam, and D Zuckerman, *Path-sampling strategies for simulating rare events in biomolecular systems*, Current Opinion in Structural Biology **43** (2017), 88–94.

[29] G Ciccotti, M Ferrario, and C Schuette, *Molecular dynamics simulation*, Entropy **16** (2014), 233.

[30] B Cook and A Podelski, *Verification, model checking, and abstract interpretation: 8th international conference, vmcai 2007, nice, france, january 14-16, 2007, proceedings*, vol. 4349, Springer, 2007.

[31] H Cramér, *Historical review of filip lundberg's works on risk theory*, Scandinavian Actuarial Journal **1969** (1969), no. sup3, 6–12.

[32] ———, *A century with probability theory: Some personal recollections*, The Annals of Probability **4** (1976), no. 4, 509–546.

[33] C De Bacco, E Power, D Larremore, and C Moore, *Community detection, link prediction, and layer interdependence in multilayer networks*, Physical Review E **95** (2017), no. 4, 042317.

[34] B Derrida, *An exactly soluble non-equilibrium system: the asymmetric simple exclusion process*, Physics Reports **301** (1998), no. 1-3, 65–83.

[35] ———, *Non-equilibrium steady states: fluctuations and large deviations of the density and of the current*, Journal of Statistical Mechanics: Theory and Experiment **2007** (2007), no. 07, P07023.

[36] B Derrida and J Lebowitz, *Exact large deviation function in the asymmetric exclusion process*, Physical Review Letters **80** (1998), 209–213.

[37] B Derrida, J Lebowitz, and E Speer, *Large deviation of the density profile in the steady state of the open symmetric simple exclusion process*, Journal of Statistical Physics **107** (2002), no. 3-4, 599–634.

[38] ———, *Exact large deviation functional of a stationary open driven diffusive system: the asymmetric exclusion process*, Journal of Statistical Physics **110** (2003), no. 3-6, 775–810.

[39] J Dongarra, S Otto, M Snir, and D Walker, *An introduction to the mpi standard*, Communications of the ACM (1995), 18.

[40] D Easterling, G Meehl, C Parmesan, S Changnon, T Karl, and L Mearns, *Climate extremes: observations, modeling, and impacts*, Science **289** (2000), no. 5487, 2068–2074.

[41] R Ellis, *Entropy, large deviations, and statistical mechanics*, Springer, 2007.

[42] C Enaud and B Derrida, *Large deviation functional of the weakly asymmetric exclusion process*, Journal of Statistical Physics **114** (2004), no. 3-4, 537–562.

[43] M Evans, Y Kafri, E Levine, and D Mukamel, *Phase transition in a non-conserving driven diffusive system*, Journal of Physics A: Mathematical and General **35** (2002), no. 29, L433.

[44] P Ferrari, E Presutti, E Scacciatelli, and M Vares, *The symmetric simple exclusion process, i: Probability estimates*, Stochastic Processes and their Applications **39** (1991), no. 1, 89–105.

[45] ———, *The symmetric simple exclusion process, ii: Applications*, Stochastic Processes and their Applications **39** (1991), no. 1, 107–115.

[46] J Garrahan, *Aspects of non-equilibrium in classical and quantum systems: Slow relaxation and glasses, dynamical large deviations, quantum non-ergodicity, and open quantum dynamics*, Physica A: Statistical Mechanics and its Applications **504** (2018), 130–154.

[47] J Garrahan, R Jack, V Lecomte, E Pitard, K van Duijvendijk, and F van Wijland, *Dynamical first-order phase transition in kinetically constrained models of glasses*, Physical Review Letters **98** (2007), no. 19, 195702.

[48] ———, *First-order dynamical phase transition in models of glasses: an approach based on ensembles of histories*, Journal of Physics A: Mathematical and Theoretical **42** (2009), no. 7, 075007.

[49] C Giardinà, J Kurchan, V Lecomte, and J Tailleur, *Simulating rare events in dynamical processes*, Journal of Statistical Physics **145** (2011), no. 4, 787–811.

[50] C Giardinà, J Kurchan, and L Peliti, *Direct evaluation of large-deviation functions*, Physical Review Letters **96** (2006), 120603.

[51] P Grassberger, *Go with the winners: A general monte carlo strategy*, Computer Physics Communications **147** (2002), no. 1, 64–70.

[52] P Gretener, *Significance of the rare event in geology*, AAPG Bulletin **51** (1967), no. 11, 2197–2206.

[53] N Gromov and V Kazakov, *Double scaling and finite size corrections in sl (2) spin chain*, Nuclear Physics B **736** (2006), no. 3, 199–224.

[54] W Gropp, W Gropp, E Lusk, and A Skjellum, *Using mpi: portable parallel programming with the message-passing interface*, vol. 1, MIT press, 1999.

[55] W Gropp and R Thakur, *An evaluation of implementation options for mpi one-sided communication*, European Parallel Virtual Machine/Message Passing Interface Users' Group Meeting, Springer, 2005, pp. 415–424.

[56] R Harris, *Fluctuations in interacting particle systems with memory*, Journal of Statistical Mechanics: Theory and Experiment **2015** (2015), no. 7, P07021.

[57] R Harris, A Rákos, and G Schütz, *Current fluctuations in the zero-range process with open boundaries*, Journal of Statistical Mechanics: Theory and Experiment **2005** (2005), no. 08, P08003.

[58] R Harris and H Touchette, *Current fluctuations in stochastic systems with long-range memory*, Journal of Physics A: Mathematical and Theoretical **42** (2009), no. 34, 342001.

[59] ———, *Large deviation approach to nonequilibrium systems*, Nonequilibrium Statistical Physics of Small Systems: Fluctuation Relations and Beyond **6** (2013), 335–360.

[60] ———, *Phase transitions in large deviations of reset processes*, Journal of Physics A: Mathematical and Theoretical **50** (2017), no. 10, 10LT01.

[61] L Hedges, R Jack, J Garrahan, and D Chandler, *Dynamic order-disorder in atomistic models of structural glass formers*, Science **323** (2009), no. 5919, 1309–1313.

[62] E Hidalgo, *Breakdown of the finite-time and-population scalings of the large deviation function in the large-size limit of a contact process*, Journal of Statistical Mechanics: Theory and Experiment **2018** (2018), no. 8, 083211.

[63] E Hidalgo and V Lecomte, *Discreteness effects in population dynamics*, Journal of Physics A: Mathematical and Theoretical **49** (2016), no. 20, 205002.

[64] E Hidalgo, T Nemoto, and V Lecomte, *Finite-time and finite-size scalings in the evaluation of large-deviation functions: Numerical approach in continuous time*, Physical Review E **95** (2017), no. 6, 062134.

[65] C Hu and K Mak, *Percolation and phase transitions of hard-core particles on lattices: Monte carlo approach*, Physical Review B **39** (1989), no. 4, 2948.

[66] G Huber and S Kim, *Weighted-ensemble brownian dynamics simulations for protein association reactions.*, Biophysical Journal **70** (1996), no. 1, 97.

[67] P Hurtado, C Espigares, J del Pozo, and P Garrido, *Thermodynamics of currents in nonequilibrium diffusive systems: theory and simulation*, Journal of Statistical Physics **154** (2014), no. 1-2, 214–264.

[68] P Hurtado and P Garrido, *Current fluctuations and statistics during a large deviation event in an exactly solvable transport model*, Journal of Statistical Mechanics: Theory and Experiment **2009** (2009), no. 02, P02032.

[69] R Jack, J Garrahan, and D Chandler, *Space-time thermodynamics and subsystem observables in a kinetically constrained model of glassy materials*, The Journal of Chemical Physics **125** (2006), no. 18, 184509.

[70] R Jack and P Sollich, *Large deviations and ensembles of trajectories in stochastic models*, Progress of Theoretical Physics Supplement **184** (2010), 304–317.

[71] ———, *Effective interactions and large deviations in stochastic processes*, The European Physical Journal Special Topics **224** (2015), no. 12, 2351–2367.

[72] J Jaeger, E Saillard, P Carribault, and D Barthou, *Correctness analysis of mpi-3 non-blocking communications in parcoach*, Proceedings of the 22nd European MPI Users' Group Meeting, ACM, 2015, p. 16.

[73] S Janson, *Brownian excursion area, wright's constants in graph enumeration, and other brownian areas*, Probability Surveys **4** (2007), 80–145.

[74] T Johnson, S Clark, and D Jaksch, *Dynamical simulations of classical stochastic systems using matrix product states*, Physical Review E **82** (2010), no. 3, 036702.

[75] T Johnson, T Elliott, S Clark, and D Jaksch, *Capturing exponential variance using polynomial resources: Applying tensor networks to nonequilibrium stochastic processes*, Physical Review Letters **114** (2015), no. 9, 090602.

[76] H Jung, K Okazaki, and G Hummer, *Transition path sampling of rare events by shooting from the top*, The Journal of Chemical Physics **147** (2017), no. 15, 152716.

[77] A Keys, L Hedges, J Garrahan, S Glotzer, and D Chandler, *Excitations are localized and relaxation is hierarchical in glass-forming liquids*, Physical Review X **1** (2011), no. 2, 021013.

[78] A Kierzek, *Stocks: Stochastic kinetic simulations of biochemical systems with gillespie algorithm*, Bioinformatics **18** (2002), no. 3, 470–481.

[79] C Kipnis and C Landim, *Scaling limits of interacting particle systems*, vol. 320, Springer Science & Business Media, 2013.

[80] C Kipnis, S Olla, and S Varadhan, *Hydrodynamics and large deviation for simple exclusion processes*, Communications on Pure and Applied Mathematics **42** (1989), no. 2, 115–137.

[81] V Kishore, M Santhanam, and R Amritkar, *Extreme events on complex networks*, Physical Review Letters **106** (2011), no. 18, 188701.

[82] I Lawrie, *Phase transitions*, Contemporary Physics **28** (1987), no. 6, 599–601.

[83] J Lebowitz and H Spohn, *A gallavotti–cohen-type symmetry in the large deviation functional for stochastic dynamics*, Journal of Statistical Physics **95** (1999), no. 1-2, 333–365.

[84] V Lecomte, C Appert-Rolland, and F Van Wijland, *Thermodynamic formalism for systems with markov dynamics*, Journal of Statistical Physics **127** (2007), no. 1, 51–106.

[85] V Lecomte, J Garrahan, and F van Wijland, *Inactive dynamical phase of a symmetric exclusion process on a ring*, Journal of Physics A: Mathematical and Theoretical **45** (2012), no. 17, 175001.

[86] V Lecomte and J Tailleur, *A numerical approach to large deviations in continuous time*, Journal of Statistical Mechanics: Theory and Experiment **2007** (2007), no. 03, P03004.

[87] T Liggett, *Stochastic interacting systems: contact, voter and exclusion processes*, vol. 324, springer science & Business Media, 2013.

[88] A Linde, *Phase transitions in gauge theories and cosmology*, Reports on Progress in Physics **42** (1979), no. 3, 389.

[89] J Marro and R Dickman, *Nonequilibrium phase transitions in lattice models*, Cambridge University Press, 2005.

[90] N McNew, *The eigenvalue gap and mixing time*, (2011).

[91] A Mey, P Geissler, and J Garrahan, *Rare-event trajectory ensemble analysis reveals metastable dynamical phases in lattice proteins*, Physical Review E **89** (2014), no. 3, 032109.

[92] P Mörters, *Introduction to large deviations*, October 19th (2010).

[93] R Movassagh, *The gap of fredkin quantum spin chain is polynomially small*, Math. Sci. Appl (2016).

[94] T Nemoto, F Bouchet, R Jack, and V Lecomte, *Population-dynamics method with a multicanonical feedback control*, Physical Review E **93** (2016), no. 6, 062123.

[95] T Nemoto, E Hidalgo, and V Lecomte, *Finite-time and finite-size scalings in the evaluation of large-deviation functions: Analytical study using a birth-death process*, Physical Review E **95** (2017), no. 1, 012102.

[96] T Nemoto, R Jack, and V Lecomte, *Finite-size scaling of a first-order dynamical phase transition: Adaptive population dynamics and an effective model*, Physical Review Letters **118** (2017), no. 11, 115702.

[97] D Nickelsen and H Touchette, *Anomalous scaling of dynamical large deviations*, Physical Review Letters **121** (2018), 090602.

[98] M Onorato, A Osborne, and M Serio, *Extreme wave events in directional, random oceanic sea states*, Physics of Fluids **14** (2002), no. 4, L25–L28.

[99] R Patel, J Ho, F Ferreyrol, T Ralph, and G Pryde, *A quantum fredkin gate*, Science Advances **2** (2016), no. 3, e1501531.

[100] H Pham, *Some applications and methods of large deviations in finance and insurance*, Paris-Princeton Lectures on Mathematical Finance 2004, Springer, 2007, pp. 191–244.

[101] E Pitard, V Lecomte, and F Van Wijland, *Dynamic transition in an atomic glass former: A molecular-dynamics evidence*, EPL (Europhysics Letters) **96** (2011), no. 5, 56002.

[102] V Popkov, L Santen, A Schadschneider, and G Schütz, *Empirical evidence for a boundary-induced nonequilibrium phase transition*, Journal of Physics A: Mathematical and General **34** (2001), no. 6, L45.

[103] U Ray, G Chan, and D Limmer, *Importance sampling large deviations in nonequilibrium steady states. i*, The Journal of Chemical Physics **148** (2018), no. 12, 124120.

[104] T Saif and M Parashar, *Understanding the behavior and performance of non-blocking communications in mpi*, European Conference on Parallel Processing, Springer, 2004, pp. 173–182.

[105] O Salberger and V Korepin, *Fredkin spin chain*, Ludwig Faddeev Memorial Volume: A Life In Mathematical Physics (2018), 439.

[106] H Shoukourian, T Wilde, A Auweter, and A Bode, *Predicting the energy and power consumption of strong and weak scaling hpc applications*, Supercomputing Frontiers and Innovations **1** (2014), no. 2, 20–41.

[107] S Siegel, *Model checking nonblocking mpi programs*, International Workshop on Verification, Model Checking, and Abstract Interpretation, Springer, 2007, pp. 44–58.

[108] M Snir, S Otto, S Huss-Lederman, J Dongarra, and D Walker, *Mpi–the complete reference: The mpi core*, vol. 1, MIT press, 1998.

[109] P Sollich and R Jack, *Large deviations of the dynamical activity in the east model: analysing structure in biased trajectories*, J. Phys. A **47** (2014), no. 015003.

[110] T Speck, A Malins, and C Royall, *First-order phase transition in a model glass former: Coupling of local structure and dynamics*, Physical Review Letters **109** (2012), no. 19, 195703.

[111] H Spohn, *Large scale dynamics of interacting particles*, Springer Science & Business Media, 2012.

[112] J Tailleur and J Kurchan, *Probing rare physical trajectories with lyapunov weighted dynamics*, Nature Physics **3** (2007), no. 3, 203–207.

[113] J Tailleur and V Lecomte, *Simulation of large deviation functions using population dynamics*, AIP Conference Proceedings **1091** (2009), no. 1, 212–219.

[114] K Temme, M Wolf, and F Verstraete, *Stochastic exclusion processes versus coherent transport*, New Journal of Physics **14** (2012), no. 7, 075004.

[115] I Thompson and R Jack, *Dynamical phase transitions in one-dimensional hard-particle systems*, Physical Review E **92** (2015), no. 5, 052115.

[116] H Touchette, *The large deviation approach to statistical mechanics*, Physics Reports **478** (2009), no. 1-3, 1–69.

[117] ———, *A basic introduction to large deviations: Theory, applications, simulations*, Tech. report, 2011.

[118] A Tzella and J Vanneste, *Dispersion in rectangular networks: effective diffusivity and large-deviation rate function*, Physical Review Letters **117** (2016), no. 11, 114501.

[119] P Vekilov, *Phase transitions of folded proteins*, Soft Matter **6** (2010), no. 21, 5254–5272.

[120] J Weber, R Jack, and V Pande, *Emergence of glass-like behavior in markov state models of protein folding dynamics*, Journal of the American Chemical Society **135** (2013), no. 15, 5501–5504.

[121] J Weber, R Jack, C Schwantes, and V Pande, *Dynamical phase transitions reveal amyloid-like states on protein folding landscapes*, Biophysical Journal **107** (2014), no. 4, 974–982.

[122] B Zhang, D Jasnow, and D Zuckerman, *The "weighted ensemble" path sampling method is statistically exact for a broad class of stochastic processes and binning procedures*, The Journal of Chemical Physics **132** (2010), no. 5, 054107.